UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

DEPARTAMENTO DE INGENIERÍA DE SISTEMAS Y AUTOMÁTICA

PROYECTO FIN DE CARRERA

INGENIERÍA INDUSTRIAL

ESPECIALIDAD AUTOMÁTICA Y ELECTRÓNICA INDUSTRIAL

"LABORATORIOS REMOTOS MEDIANTE MATLAB WEB SERVER Y EASY JAVA SIMULATIONS"

Autor: Rubén Darío Crespo Sánchez

Tutor: D. Ramón Ignacio Barber Castaño

Octubre, 2011

Índice

Índice de figuras

Índice de ecuaciones

Resumen

En el presente proyecto se realiza un estudio de diferentes tecnologías para implementar laboratorios remotos.

La finalidad es que los alumnos puedan realizar prácticas de asignaturas de ingeniería de control desde cualquier lugar en el que dispongan de un PC con conexión a internet.

Este proyecto surge debido al gran número de estudiantes que existe y el reducido número de sistemas de laboratorio, sobre todo cuando se trata de sistemas de laboratorio específicos que tienen un coste realmente elevado.

Además, con este sistema se puede ampliar el tiempo que el alumno puede dedicar a las prácticas con el consiguiente beneficio para su aprendizaje que esto supone.

El alumno sólo precisa de un PC con conexión a internet, un navegador web y Java instalados. En el laboratorio se precisa un PC con conexión a internet, un servidor Apache instalado, MATLAB instalado incluyendo MATLAB Web Server, Simulink y Data Acquisition Toolbox.

Se estudian dos tecnologías diferentes. Por un lado está MATLAB Web Server y por otro Easy Java Simulations. Dentro de cada tecnología se han realizado diferentes opciones.

Con MATLAB Web Server es posible simular un modelo de Simulink de forma remota que muestre los resultados mediante una tabla o en una gráfica. También es posible introducir datos del modelo de Simulink a simular de forma remota mostrando los resultados gráficamente. Finalmente también se puede conectar con un motor real de forma remota mostrando los resultados gráficamente.

Con Easy Java Simulations se puede simular un modelo de Simulink y conectarse con un motor real de forma remota mostrando los resultados gráficamente.

En conclusión, este proyecto es la base de cómo configurar y usar estas tecnologías para realizar prácticas de forma remota. Se pueden incluir tanta variedad de prácticas de MATLAB como el profesor quiera.

Capítulo 1

Introducción

1 Introducción

En el presente proyecto se ha creado un sistema de laboratorio remoto para que los alumnos puedan realizar prácticas desde cualquier lugar gracias a internet. En enseñanzas como las ingenierías es fundamental que el alumno además de los contenidos teóricos realice prácticas con equipos reales para poder entender y aplicar los contenidos teóricos. Por ello ya son varias las universidades que han desarrollado laboratorios remotos. Lo que se pretende en este proyecto es que el alumno pueda realizar una práctica como si estuviera en el laboratorio, conectarse con el equipo real y después analizar los resultados, pero desde casa.

El sistema sobre el que se ha probado este proyecto es la misma maqueta que se usa en el Departamento de Ingeniería de Sistemas y Automática de la Universidad Carlos III de Madrid en las prácticas de asignaturas de ingeniería de control. Esta maqueta está compuesta por un motor de corriente continua controlado por armadura, con velocidad de 4000RPM a 24Vcc, y que incorpora una tacogeneratriz como sensor de velocidad y un encoder como sensor de posición. La maqueta tiene unos conectores por los que se conecta el motor a un PC mediante una tarjeta de adquisición de datos, en concreto la tarjeta de adquisición de datos "Advantech PCI-1711". En la Figura 1.1 se puede observar la maqueta del laboratorio con el motor en la parte de la izquierda, la tarjeta de adquisición de datos en la parte de la derecha y las conexiones en el centro.

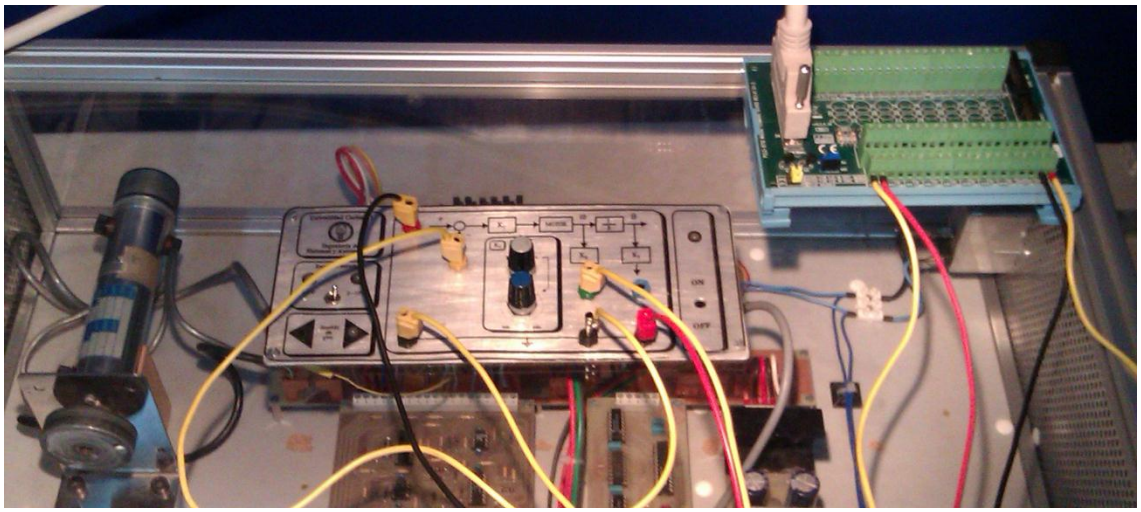


Figura 1.1. Sistema real del laboratorio

A continuación se van a detallar los motivos para realizar este proyecto y los objetivos que se querían alcanzar.

1.1 Motivación

La principal motivación de este proyecto se basa en la limitada disponibilidad de sistemas físicos reales para la realización de prácticas de control y del tiempo limitado de acceso a los laboratorios. Pero existen más motivaciones, como que las licencias de MATLAB tienen un precio muy elevado, factor que hace que los alumnos no puedan adquirirlas y no puedan realizar simulaciones desde sus casas.

Las prácticas son algo imprescindible en enseñanzas técnicas, sin embargo la disponibilidad de sistemas físicos hace que no se dediquen las horas que serían necesarias. Con un laboratorio remoto se consigue que el alumno pueda realizar más prácticas y con más tiempo, por lo que conseguirá aplicar más conceptos teóricos y aprender más. La motivación de que los alumnos puedan aprender más es muy importante, ya que hay que buscar cómo conseguir que los alumnos se preparen adecuadamente.

1.2 Objetivos

El objetivo que se persigue en este proyecto es que los alumnos puedan realizar prácticas de forma remota, bien sea sobre un sistema físico, o simplemente mediante simulaciones con Simulink. Así se consigue dar más disponibilidad al sistema físico ya que puede trabajar 24h al día. Los alumnos pueden practicar más, por lo tanto aprender más. Los alumnos no tienen por qué gastar en licencias de MATLAB, les basta con conectarse a la página de la asignatura y practicar.

Los objetivos propuestos para este proyecto son:

- Realizar un estudio de diferentes tecnologías para implementar laboratorios remotos.
- Implementar una solución basada en MATLAB Web Server.
- Como objetivo extra y en función de los resultados que se iban obteniendo con MATLAB Web Server, se decidió implementar otra solución con Easy Java Simulations.

Estos objetivos se definen antes de empezar todo proyecto. No obstante, en determinadas ocasiones no se consiguen los objetivos planificados. En este proyecto se han conseguido todos los objetivos planificados, aunque no por el método que se esperaba conseguir. Ha sido necesario investigar otras opciones diferentes, por lo que se han usado tecnologías y herramientas que no se esperaban usar, como la tecnología Easy Java Simulations y la Data Acquisition Toolbox de MATLAB. En un principio se pretendía dar solución a los objetivos mediante MATLAB con MATLAB Web Server y

Simulink solamente. Este proyecto es un proyecto de investigación en el que se han probado muchas posibilidades hasta lograr los objetivos buscados. De ahí que la interfaz gráfica no esté muy cuidada. Sin embargo, el funcionamiento es correcto que es lo que se buscaba en este proyecto.

1.3 Partes del documento

En este apartado se da una visión global del documento especificando de qué consta cada parte.

En primer lugar se ha realizado un breve resumen del presente proyecto final de carrera con el que se consigue una visión general de lo que trata este proyecto.

En segundo lugar se especifica con más detalle los motivos para realizar este proyecto, así como los objetivos a conseguir.

Después de esto, se detalla el estado del arte de lo que se pretende realizar, es decir, recopilación de información de los diferentes sistemas de laboratorios remotos que han creado otras universidades clasificados en función del tipo de tecnología que se ha usado para conseguir ese fin.

En el siguiente capítulo se explica la principal solución a este proyecto. Es la solución con MATLAB Web Server. Se dan todo tipo de detalles sobre cómo configurar desde cero un sistema para conseguir que tenga una funcionalidad completa. Desde la configuración de MATLAB y Apache, pasando por cómo se han creado los sistemas que se han usado, hasta la explicación de las diferentes conexiones.

Una vez explicada la solución principal, también se explica otra solución que se ha encontrado. Es la solución con Easy Java Simulations. Se explican todos los detalles sobre cómo configurar los diferentes modos de conexión.

Después de explicar los detalles para poner en funcionamiento el sistema, se procede a la validación mediante su implementación y pruebas correspondientes. Se aportan abundantes imágenes que permiten avalar el buen funcionamiento de los sistemas propuestos. Además se detalla una primera implementación de la primera práctica de una asignatura, guiando paso a paso al alumno.

Finalmente se da una conclusión del proyecto en general realizando diferentes reflexiones. También se proponen varios trabajos futuros para este proyecto que por falta de tiempo no se han podido desarrollar, y otros trabajos que harían de este proyecto una gran ayuda para los alumnos.

Después de esto se hace referencia a las fuentes usadas para el desarrollo y escritura de este proyecto.

Antes de incluir la información adjunta necesaria para la comprensión del proyecto se detalla el presupuesto de este proyecto.

Finalmente se adjunta cierta información que no corresponde incluirla en la redacción del proyecto, sin embargo es una información con suficiente importancia como para aparecer en el anexo.

Capítulo 2

Sistemas de laboratorios remotos

2 Sistemas de laboratorios remotos

En la actualidad existen diferentes universidades que han creado sistemas de laboratorios remotos. Estas universidades han usado diferentes tecnologías. A continuación se explican algunas de estas tecnologías, y cómo han llegado a una solución.

2.1 Soluciones con MATLAB Web Server

MATLAB hace unos años incorporaba una toolbox llamada MATLAB Web Server. Esta toolbox permitía conectarse con MATLAB de manera remota desde una página HTML. Incluso se podía simular un modelo de Simulink. El funcionamiento se basa en enviar a través de un formulario HTML los datos necesarios. MATLAB o Simulink realizan los cálculos precisos y devuelven los resultados de los cálculos realizados o de la simulación vía HTML. Es una toolbox bastante útil, pero con ciertas limitaciones. Estas limitaciones son el motivo por el que retiraron esta toolbox de MATLAB.

La limitación más importante que tiene MATLAB Web Server es que no puede ejecutar un modelo de Simulink en modo externo, sólo puede ejecutar modelos de Simulink en modo normal, por lo que es no es posible conectarse a una planta remotamente mediante Simulink, ya que para ello se necesita el modo externo. Esta toolbox de MATLAB está mayormente destinada al uso de laboratorios virtuales, pero no al uso de laboratorios remotos ya que no puede conectarse con una planta real mediante Simulink. Las soluciones que se van a estudiar usan MATLAB Web Server, solventando la limitación que tiene de diferentes modos. El uso de esta toolbox queda reducido a adaptar los datos a MATLAB y a la generación de páginas web en HTML. En este proyecto se aporta otra solución diferente que solventa esta limitación.

Una de las soluciones es “Recolab: Laboratorio remoto de control utilizando MATLAB y Simulink” (Luis M. Jiménez, Rafael Puerto, Óscar Reinoso, César Fernández, Ramón Ñeco, 2005) creado en la Universidad de Miguel Hernández (Alicante). En esta solución se usa MATLAB Web Server para adaptar los datos a MATLAB. El resto de tareas se han desarrollado en PHP usando ficheros para comunicar los datos y recuperar los resultados obtenidos del experimento realizado. Es una solución muy elaborada, con una alta seguridad. En la Figura 2.1 se puede observar una página de resultados de esta tecnología.

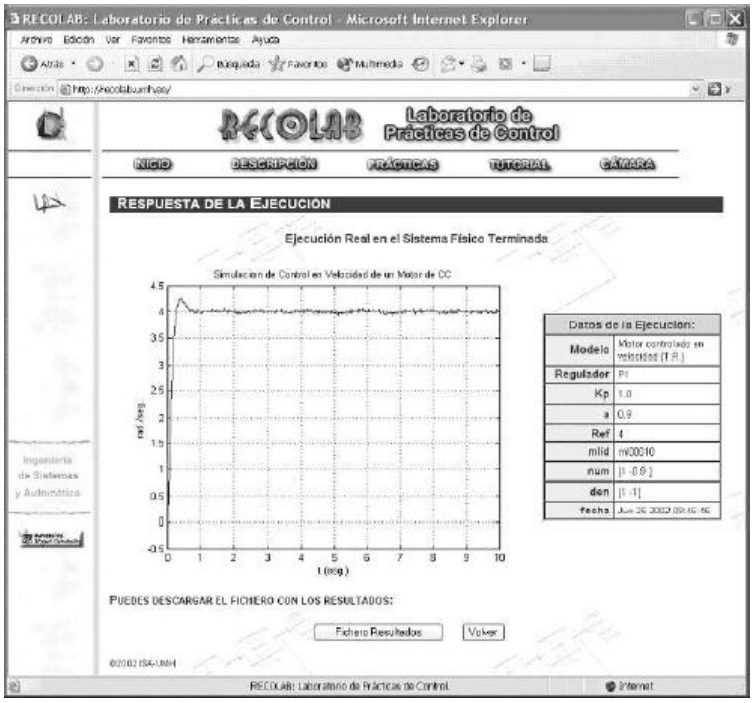


Figura 2.1. Interfaz gráfica de la solución Recolab

Otra solución de la Universidad de Miguel Hernández (Alicante) es "Remote lab for control applications usign MATLAB" (Rafael Puerto, Óscar Reinoso, Ramón Ñeco, Nicolás García, Luis M. Jiménez, 2001). En esta opción, se deja un programa de MATLAB corriendo en un bucle infinito que espera a que esté disponible un fichero creado a partir de los datos recibido por MATLAB Web Server. Una vez que se tiene el fichero, MATLAB realiza las operaciones necesarias en el sistema en tiempo real y al finalizar se genera otro fichero con los resultados que será leído mediante MATLAB Web Server. Esta solución no es tan elaborada como la otra que tienen en esta universidad, pero tiene una funcionalidad básica.

Program executed by Matlab Web Server

```
Read data from user
Testing the controller
If (Controller OK) Then
   Generate data file
   While not exist results file
      %Loop
   end
   Load results file
   Delete results file
   Generate results from user

Else
   Generate error message from user
```

Program executed by Matlab

```
While True % Infinite Loop
  While not exist data file
      %Loop
  End
  Load data file
  %Real time execution
  Connect to Real - time Kernel
  Start Simulink Model execution
     %Real Time Execution
  Stop Simulink Model execution
  Disconnect from Real - Time Kernel
  Save results file
  Delete data file
end
```

Figura 2.2. Pseudocódigo de MATLAB y MATLAB Web Server

Como se puede observar en el pseudocódigo de la Figura 2.2, no es una solución muy elaborada, ya que deja MATLAB corriendo en un bucle infinito funcionando continuamente en el PC remoto. En cuanto que ese PC tuviera algún error, podría cancelarse el proceso y sería necesario arrancar de nuevo MATLAB.

Existe una solución de la Universidad Politécnica de Valencia: "Simulación y control de procesos físicos de forma remota" (A. Valera, M. Vallés, J.L. Díez, 2005). Aquí se solventa la limitación de MATLAB Web Server mediante funciones software para el acceso a la tarjeta de adquisición de datos, o mediante otra aplicación, como WinCon. En la Figura 2.3 se puede observar la interfaz gráfica de esta solución.

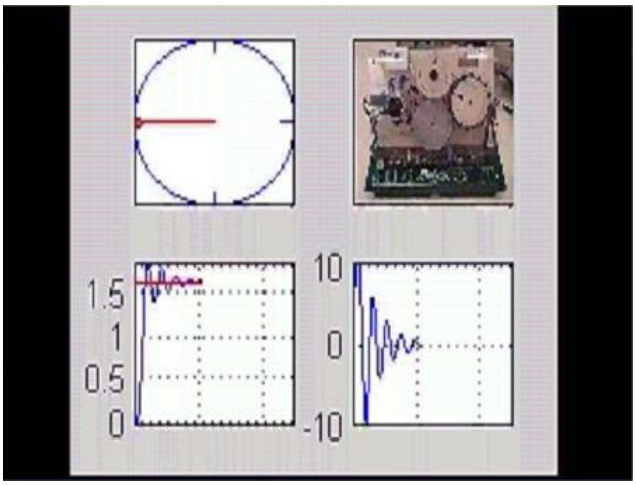


Figura 2.3. Interfaz gráfica de la solución de la Universidad Politécnica de Valencia

2.2 Soluciones con Easy Java Simulations

Existen varias soluciones de sistemas de laboratorios remotos desarrolladas con Easy Java Simulations.

Una de las soluciones es "Developing Networked Control Labs: A MATLAB and Easy Java Simulations Approach" (Gonzalo Farias, Robin De Keyser, Sebastián Dormido, Francisco Esquembre, 2010) de la Universidad Nacional de Educación a Distancia. En esta solución no usan Simulink, simplemente se conecta a la planta real mediante MATLAB.

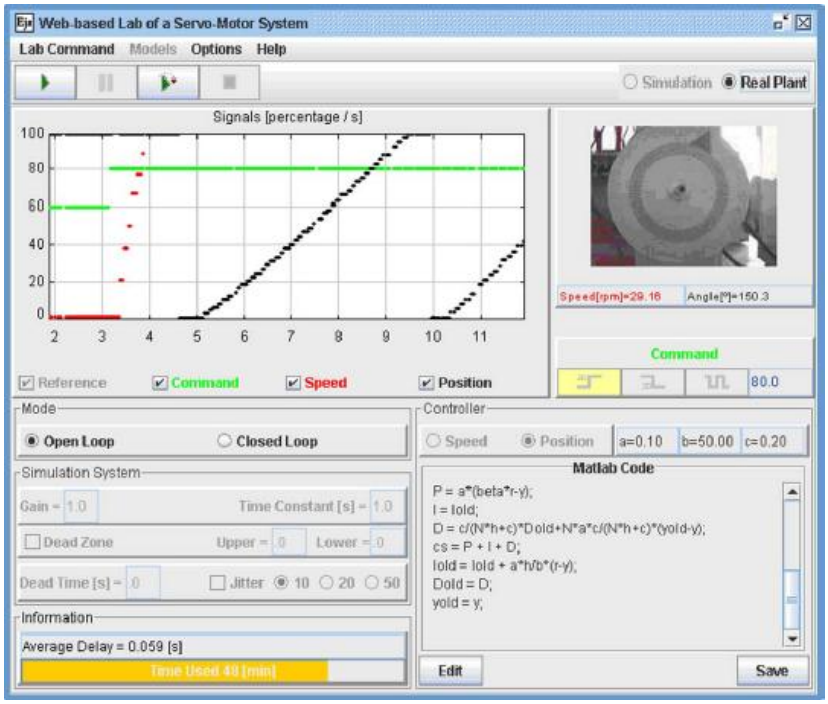


Figura 2.4. Interfaz gráfica de una de las soluciones de la UNED

La otra solución es "Web-Based virtual lab and remote experimentation using Easy Java Simulations" (R. Pastor, J. Sánchez, S. Dormido, 2005), también de la Universidad Nacional de Educación a Distancia. Muy similar a la anterior, ya que hay un autor que está implicado en los dos proyectos, pero también se incluye la opción de realizar simulaciones virtuales, sin conectar a la planta real.

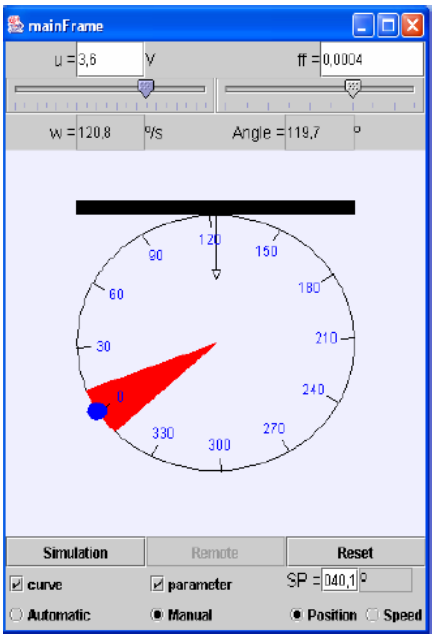


Figura 2.5. Representación virtual del motor de una de las soluciones de la UNED

En la Figura 2.5 se puede ver cómo han diseñado la representación virtual del motor sobre el cual se realiza el ensayo mediante el uso de Easy Java Simulations. Desde esa interfaz se introducen los datos a enviar al motor.

2.3 Soluciones con otro tipo de tecnologías

Existen varios sistemas de laboratorios remotos que no usan MATLAB Web Server ni Easy Java Simulations como tecnologías bases para conectarse con MATLAB. Estos sistemas usan tecnologías diferentes que también funcionan, ya tengan una complejidad mayor o menor. A continuación se detallan alguna de las otras soluciones posibles que existen.

La Universidad Pontificia de Comillas (Madrid) tiene una solución llamada: "Acceso remoto a ensayos de control en tiempo real basados en MATLAB" (Rafael Palacios, Ramón Rodríguez Pecharromán, 2005). En esta solución se ejecuta MATLAB mediante acceso remoto y utilizando instrucciones rsh (usando Services for Unix). El nombre del sistema se llama WLAB, trabajo conseguido mediante dos proyectos fin de carrera titulados "WLAB – Sistema de control remoto en tiempo real de equipos de laboratorio" (Rosa, 2006; Fernández, 2007). En la Figura 2.6 se puede observar la interfaz gráfica de esta solución.

Figura 2.6. Interfaz gráfica de la solución WLAB

Otra solución es "Laboratorio remoto de Automática: Plantas de variable continua" (David Jofre Hernández, Ramon Costa Castelló, Luis Basañez Villaluenga). La conexión de los equipos en tiempo real se realiza mediante RTLinux. El resto está implementado en Java. En la Figura 2.7 se puede observar la interfaz gráfica de esta solución. Como se puede observar es una interfaz gráfica poco elaborada. Esto se debe a lo compleja que es la solución, por lo que pudieron dedicar poco tiempo al desarrollo de la interfaz gráfica.

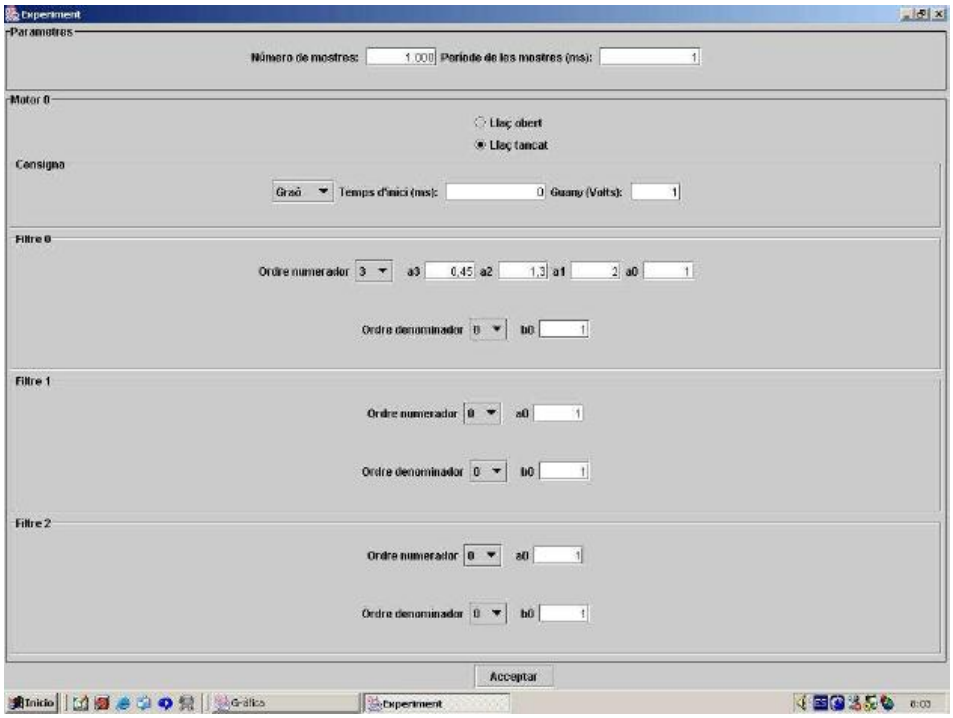


Figura 2.7. Interfaz gráfica de la solución con RTLinux

Capítulo 3

Solución con MATLAB Web Server

3 Solución con MATLAB Web Server

En este proyecto se ha encontrado una solución a los objetivos planteados usando MATLAB, en concreto usando una toolbox que facilita MATLAB, llamada MATLAB Web Server.

MATLAB (abreviatura de MATrix LABoratory) es un software matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M). Se encuentra disponible para las plataformas Unix, Windows y Apple Mac OS X.

MATLAB es un entorno de computación y desarrollo de aplicaciones totalmente integrado orientado para llevar a cabo proyectos en donde se encuentren implicados elevados cálculos matemáticos y la visualización gráfica de los mismos.

Entre sus prestaciones básicas se encuentran: la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario (GUI) y la comunicación con programas en otros lenguajes y con otros dispositivos hardware. El paquete MATLAB dispone de dos herramientas adicionales que expanden sus prestaciones: Simulink (plataforma de simulación multidominio) y GUIDE (editor de interfaces de usuario - GUI). Además, se pueden ampliar las capacidades de MATLAB con las cajas de herramientas (toolboxes) como se ha hecho en este proyecto; y las de Simulink con los paquetes de bloques (blocksets).

Las toolboxes de MATLAB extienden significativamente el número de funciones incorporadas en el programa principal. Estos toolboxes cubren casi todas las áreas principales en el mundo de la ingeniería y la simulación. Es un entorno de cálculo técnico, que se ha convertido en estándar de la industria, con capacidades no superadas en computación y visualización numérica.

Es un software ampliamente usado entre universidades y centros de investigación y desarrollo. MATLAB proporciona al ingeniero un medio de carácter único, para resolver los problemas más complejos y difíciles.

En este proyecto se usa MATLAB en general, incluyendo la herramienta de Simulink, así como la toolbox de MATLAB Web Server y la de Data Acquisition.

La toolbox de MATLAB Web Server permite crear aplicaciones de MATLAB que usen las capacidades de internet para enviar datos a MATLAB remotamente. MATLAB computa estos datos y luego los muestra mediante un explorador de internet. MATLAB Web Server usa el protocolo TCP/IP para la transmisión de datos entre el cliente y MATLAB. Es necesario que el hardware y el software de red estén instalados en el sistema antes de usar MATLAB Web Server.

En la configuración más simple, que es la usada en este proyecto, en el PC cliente se ejecuta un navegador web que no tiene por qué tener instalado MATLAB, mientras que en el PC servidor se ejecuta MATLAB, MATLAB Web Server y el servidor web (HTTPD). En la Figura 3.1 se puede observar esta configuración.

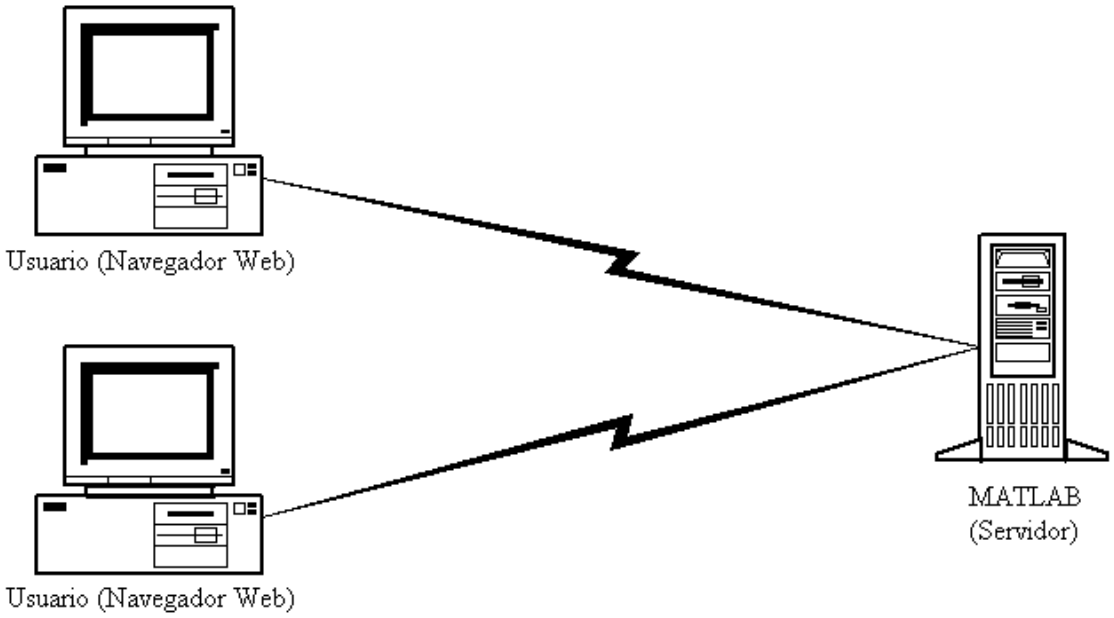


Figura 3.1. Configuración básica de MATLAB Web Server

Para poder usar MATLAB Web Server y así acceder a un PC remotamente, es necesario disponer de dos PCs mínimo. Uno será el servidor, que tendrá instalado un servidor, MATLAB con la toolbox MATLAB Web Server, Simulink y dispondrá de conexión a internet, mientras que el otro simplemente será un PC con conexión a internet y con un explorador de internet instalado.

Con el fin de que se comprenda mejor el funcionamiento del sistema está la Figura 3.2. Como se puede ver en esta figura, un usuario se conecta desde un PC local (el PC de su casa por ejemplo) a través de internet al PC remoto (el PC que está en el laboratorio). Mediante un explorador web, el usuario manda los parámetros del ensayo que quiere realizar al PC remoto. El PC remoto tiene instalado un servidor apache HTTP al igual que MATLAB, incluyendo Simulink, MATLAB Web Server toolbox y Data Acquisition toolbox. El PC remoto manda los parámetros que ha recibido a la tarjeta de adquisición de datos, que pone en funcionamiento el motor con los parámetros que le ha indicado el usuario. La tarjeta de adquisición de datos lee la salida del motor y traduce estos valores de tensión a unos valores numéricos que más tarde computará un fichero de MATLAB. Una vez se han computado los datos en un fichero de MATLAB, el PC remoto envía vía internet los resultados de la simulación al PC local. Los resultados se visualizan en el PC local mediante un explorador web. En el explorador web del usuario se mostrará una imagen del motor que captura la cámara web. La cámara web envía en todo momento los datos al PC remoto que a su vez los envía al PC local.

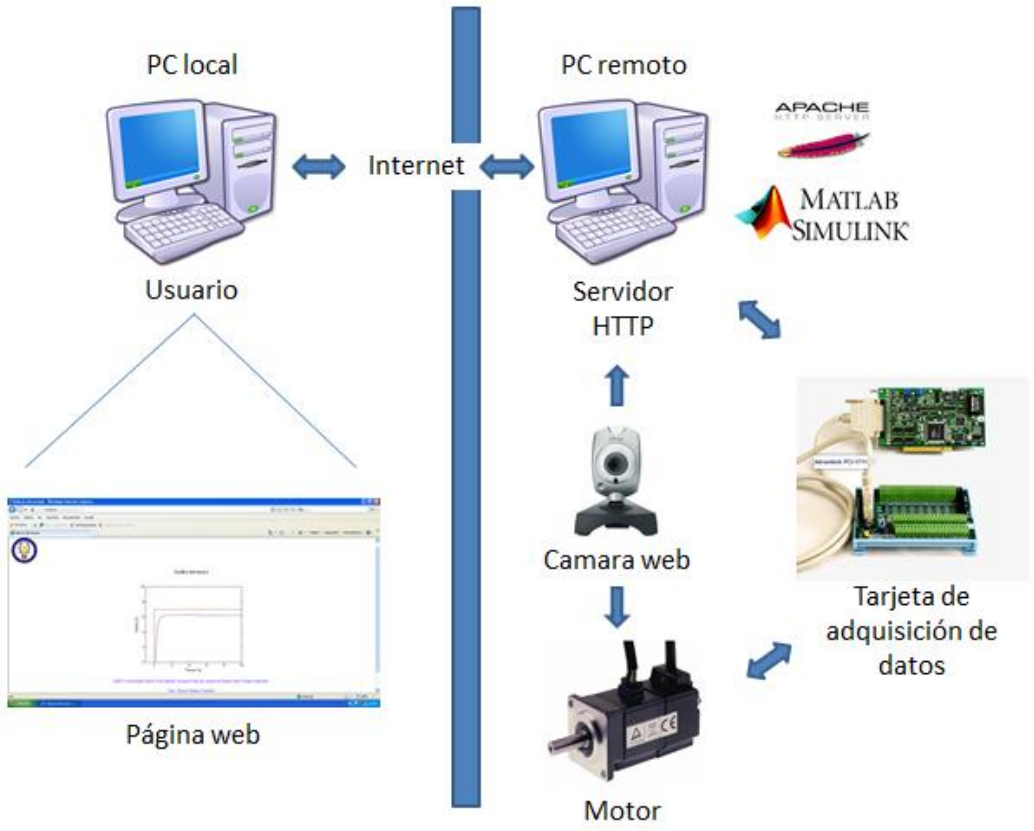


Figura 3.2. Esquema conceptual del funcionamiento del sistema

En la Figura 3.3 se muestra un esquema del funcionamiento del sistema con más detalle que el de la Figura 3.2.

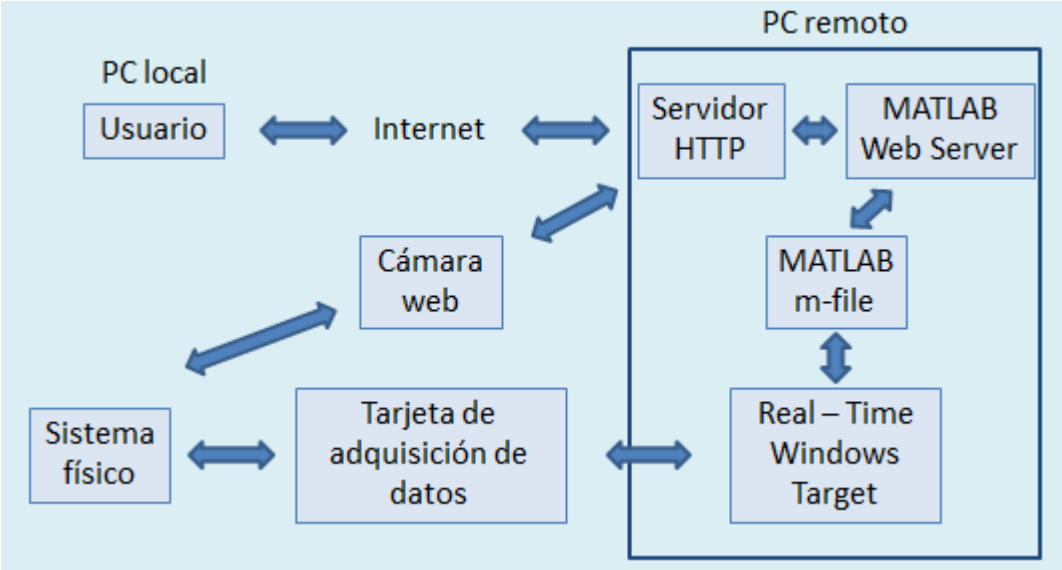


Figura 3.3. Esquema detallado del funcionamiento del sistema con MWS

En la Figura 3.4 se muestra en detalle el funcionamiento interno de MATLAB Web Server.

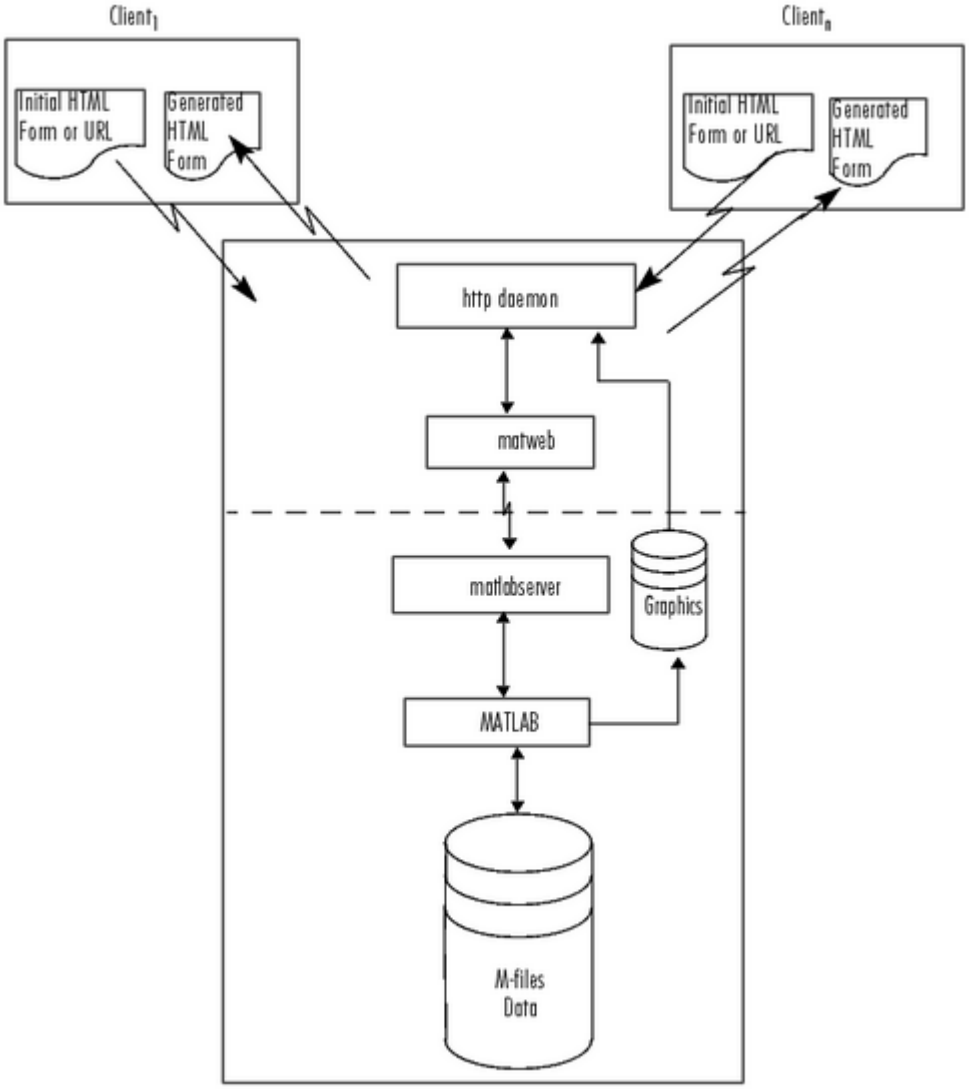


Figura 3.4. Esquema del funcionamiento de MATLAB Web Server

3.1 Configuración de MATLAB y Apache

Es necesario seguir unos ciertos pasos en la configuración de MATLAB y Apache para lograr el funcionamiento sin errores de MATLAB Web Server. Estos pasos se detallan a continuación minuciosamente:

1) Instalar MATLAB incluyendo la toolbox MATLAB Web Server en el PC que funcionará como servidor. Nótese que las versiones recientes de MATLAB ya no incluyen esta toolbox.

2) Una vez instalado MATLAB en el PC, ir a la ruta C:/MATLAB7/toolbox/webserver/wsdemos. En esta ruta existe un archivo llamado matweb.conf. Es necesario abrir dicho archivo con el bloc de notas o WordPad y reemplazar lo siguiente:

   - **<MATLABSERVER_HOST_NAME>** reemplazar por el nombre del PC.
   - **$MATLAB** por el directorio donde está instalado MATLAB. En el caso de estudio de este proyecto es: C:/MATLAB7/toolbox/webserver/wsdemos/.

   El nombre de nuestro PC se puede obtener de dos maneras: una es en Inicio, click derecho en Mi PC y en la pestaña General donde pone Registrado a nombre de: el nombre del equipo es el 2º nombre que aparece. La otra manera es poner la dirección IP que da el adaptador de red al equipo. Esta dirección se consigue escribiendo "cmd" en Ejecutar del menú Inicio, esto abre el símbolo del sistema y ahora es necesario escribir "ipconfig". Una vez pulsado "enter" después de escribir "ipconfig" la dirección del PC en cuestión es la se indica en Dirección IP. También es suficiente con escribir "localhost", ya que esto indica el nombre del PC.

3) Descargar un servidor Apache para instalarlo en el PC. Desde la página web http://www.apache.org se puede descargar el archivo para instalar el servidor Apache. Dentro de la página web hay que hacer "click" en "HTTP server" y seleccionar una de las versiones, por ejemplo descargar el archivo "apache_2.2.14-win32-x86-no_ssl.msi".

4) Instalar el servidor Apache en el PC. Durante la instalación se piden una serie de datos que es preciso ir rellenando:

   - Network Domain: **mathworks.com**
   - Server Name: **<hostname>.mathworks.com** (ejemplo: localhost.mathworks.com)
   - Administrator's email address: especificar una dirección de correo a la que puedan contactar los usuarios en caso de fallo del servidor.
   - Elegir que el servicio funcione para todos los usuarios.
   - Seleccionar instalación típica.

5) Una vez instalado el servidor Apache en el PC, ir a la ruta C:/Archivos de programa/Apache Software Foundation/Apache2.2/conf. En ese directorio hay un archivo llamado httpd.conf. Es necesario abrir dicho archivo con el bloc de notas o WordPad y escribir las siguientes instancias:

- El puerto para escuchar tiene que ser el 80. Buscar la línea de código sin comentarios (# significa comentario) donde pone Listen xx y escribir **Listen 80**
- En ServerAdmin escribir la dirección de correo electrónico. Ejemplo: **ServerAdmin usuario@usuario.com**
- En DocumentRoot hay que poner la ruta donde se encuentren los archivos que usará el servidor como las páginas HTML, los archivos de MATLAB, de Simulink, el fichero matweb.conf y su ejecutable matweb.exe. Ejemplo: **DocumentRoot "C:/MATLAB7/toolbox/ webserver/wsdemos/"**
- Es preciso crear dos directorios virtuales o alias cgi-bin e icons que apunten a la ruta C:/MATLAB7/toolbox/ webserver/wsdemos/:

```
Alias /icons/ "C:/MATLAB7/toolbox/ webserver/wsdemos/"
<Directory "C:/MATLAB7/toolbox/ webserver/wsdemos/">
   Options Indexes MultiViews
   AllowOverride None
   Order allow,deny
   Allow from all
</Directory>

ScriptAlias /cgi-bin/ "C:/MATLAB7/toolbox/ webserver/wsdemos/"
<Directory "C:/MATLAB7/toolbox/ webserver/wsdemos/">
   AllowOverride None
   Options None
   Order allow,deny
   Allow from all
</Directory>
```

Típicamente la parte de Alias no existe y es necesario escribirla justo después de donde está escrito Alias con comentario (# Alias: Maps web

paths...). En la parte de ScriptAlias sólo será necesario modificar la ruta, ya que lo demás viene escrito por defecto.

6) Después de realizar todos los pasos anteriores es necesario reiniciar el PC. Una vez el PC se ha reiniciado hay que comprobar que MATLAB Server se está ejecutando. Esto se puede comprobar introduciendo en Ejecutar "msconfig" y en la pestaña Servicios debe aparecer MATLAB Server como activo. Si aparece como detenido o no aparece es necesario reinstalar MATLAB Web Server. Después de la reinstalación y de volver a reiniciar el PC debería aparecer. Si con todo ello, sigue apareciendo como detenido, es necesario iniciar MATLAB Server haciendo "click" derecho en Mi PC/Administrar/Servicios y Aplicaciones/Servicios y buscar MATLAB Server. Desde ahí se puede iniciar o detener MATLAB Server. En el caso de que no apareciese MATLAB Server es necesario reinstalar MATLAB completamente incluyendo la toolbox MATLAB Web Server.

7) Una vez que funciona MATLAB Server hay que ir a la ruta C:/MATLAB7/webserver. En ese directorio hay un archivo llamado MATLABserver.conf. Es necesario abrir dicho archivo con el bloc de notas o WordPad y escribir: **o p 9999** (o indicar cualquier puerto que no se use).

8) En la ruta C:/MATLAB7/toolbox/webserver/wsdemos, abrir el archivo matweb.conf con el bloc de notas o WordPad y añadir en el primera línea: **mlport 9999**.

9) Una vez llegados a este punto es preciso arrancar MATLAB Server (que debería arrancar por defecto al iniciar Windows), y el servidor Apache (Inicio/Todos los programas/Apache HTTP Server 2.2/Control Apache Server/Start).

10) Para comprobar que el servidor Apache funciona correctamente es necesario abrir el explorador de internet y escribir en la barra de direcciones: http://localhost. Si todos los pasos se han seguido correctamente aparecerá una página web con las demos que ofrece MATLAB, pinchar una de ellas y comprobar que MATLAB Server funciona correctamente haciendo ejecutar algún archivo de MATLAB mediante estas demos.

11) En este momento nuestro PC es visible dentro de una red doméstica, pero si tenemos una dirección IP dinámica no podremos acceder desde el exterior introduciendo la dirección IP pública de nuestro PC. Es necesario conseguir una IP fija.

12) Una vez conseguida una dirección IP fija, para acceder desde un PC externo al nuestro hay que escribir en la barra de direcciones del explorador de internet: http://xx.xx.xxx.x donde las x representan los números de nuestra IP pública. Escribiendo esta dirección desde el mismo PC en el que está el servidor también funciona, al igual que escribiendo la IP local. No obstante, la idea de este proyecto es la realización de prácticas remotas por parte de los alumnos, por lo que se incluirá está página web dentro de la página web http://docenciaisa.uc3m.es/, para que los alumnos puedan acceder a ella fácilmente.

Nota: Todos estos pasos están referidos a la instalación de MATLAB 7.0 en el disco local C:/. En caso de que se instale otra versión de MATLAB o en otro disco local será necesario cambiar dichos nombres en todas las rutas especificadas. Por ejemplo si se usa MATLAB 6.0 y está instalado en D:/, la ruta será: D:/MATLAB6. Al igual pasa con la instalación del servidor Apache, las rutas están referidas a Apache 2.2. No obstante, es aconsejable instalar la versión 7.0 de MATLAB, así como la versión 2.2 de Apache, ya que se ha comprobado que estas versiones son compatibles y funcionan a la perfección, de todos modos, debería funcionar con otras versiones.

Una vez realizados estos pasos se podrán ver las demos que ofrece MATLAB. En la Figura 3.5 se ve la página principal en la que se puede elegir entre las cuatro demos.

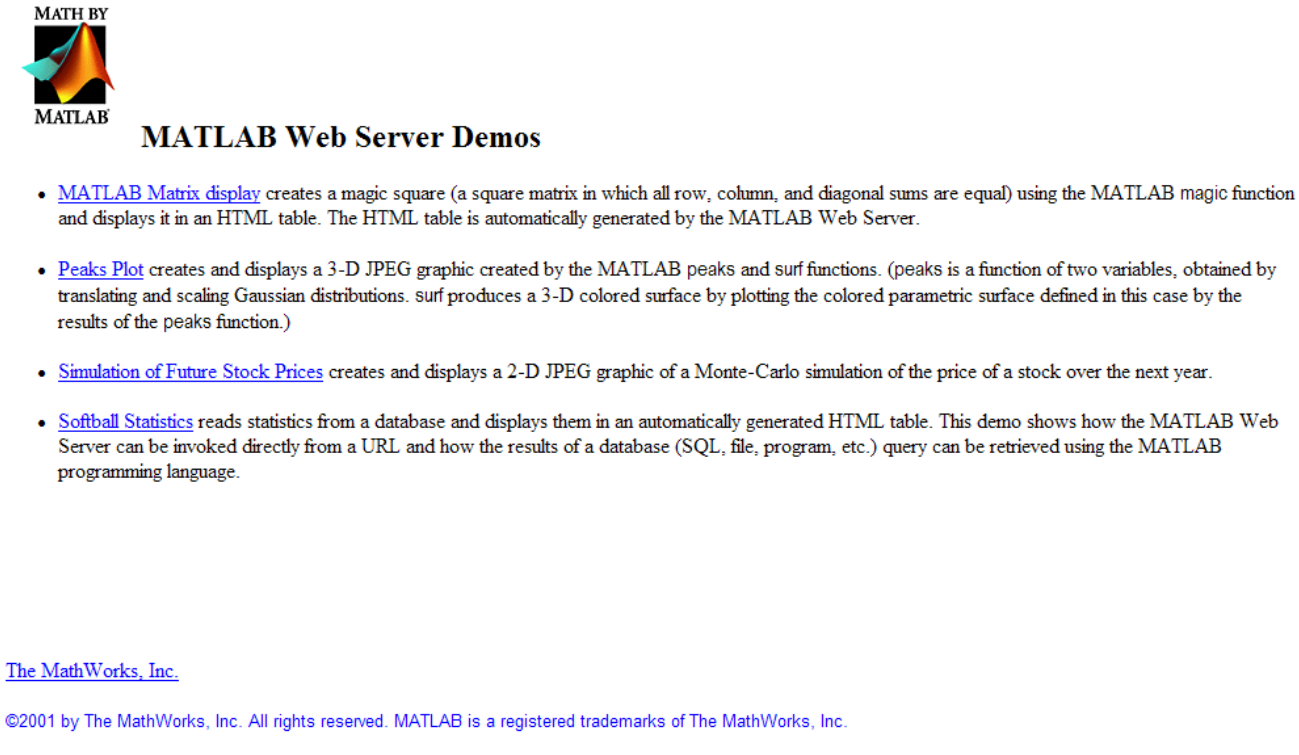


Figura 3.5. Página principal de las demos de MATLAB Web Server

Para comprobar que la configuración se ha conseguido con éxito y con el fin de comprobar también el correcto funcionamiento de MATLAB Web Server, se selecciona una de las demos, por ejemplo la primera, y aparece una página HTML con el contenido de la Figura 3.6. Una vez que se introduce un dato, por ejemplo el número "5" y se pulsa el botón "Submit", aparece la salida de datos (ver Figura 3.7).

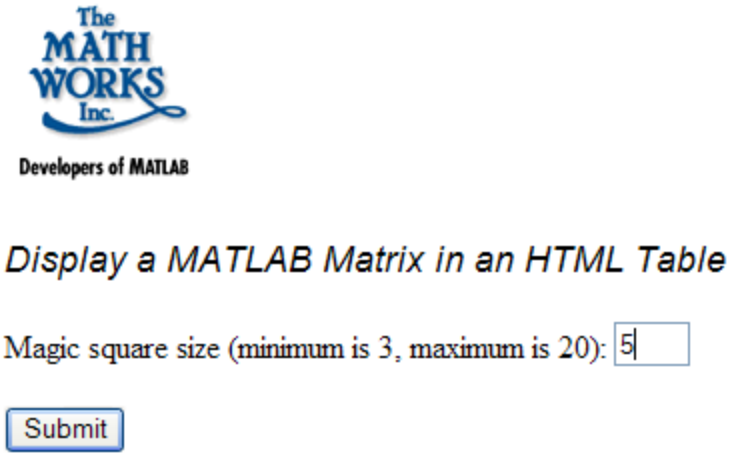


Figura 3.6. Entrada de datos del primer ejemplo

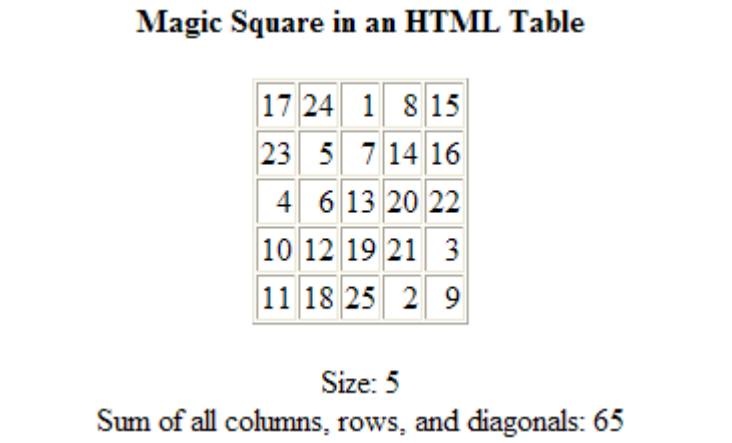


Figura 3.7. Salida de datos del primer ejemplo

En la Figura 3.8 se puede observar otro de los ejemplos de MATLAB Web Server, que muestra los resultados en la misma página donde se introducen los datos.

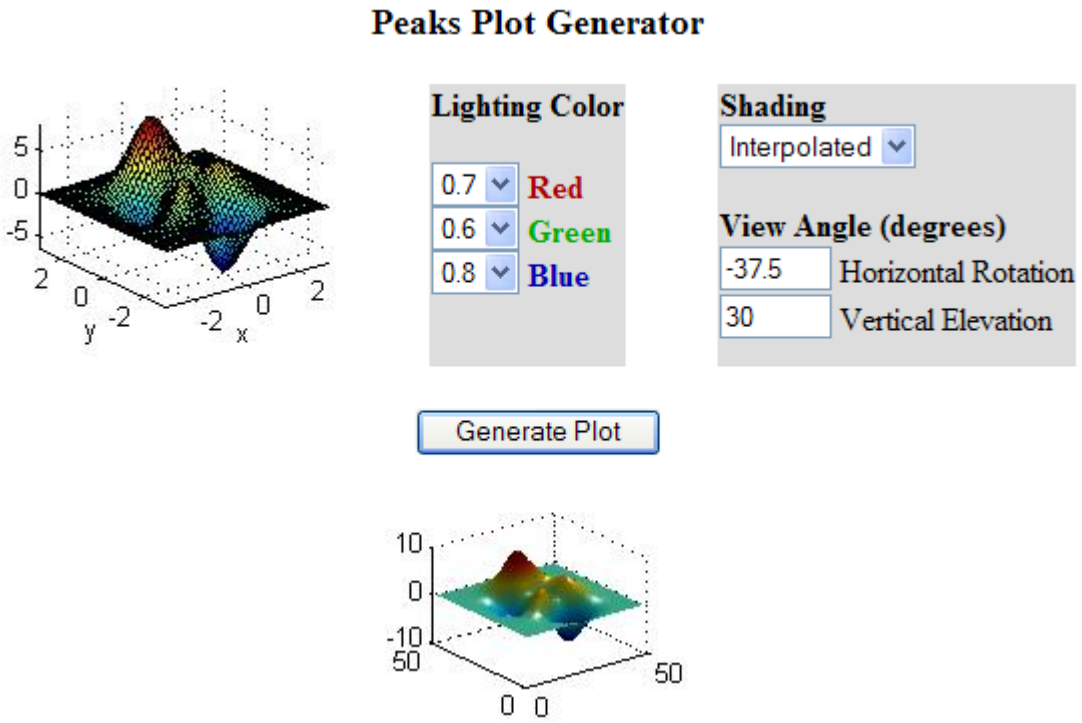


Figura 3.8. Segundo ejemplo de MATLAB Web Server

3.2 Sistemas simulados

Como se ha visto mediante las figuras del apartado anterior, los pasos indicados sirven para configurar y hacer funcionar el sistema remotamente con las demos suministradas por MATLAB.

El objeto de este proyecto no es usar las demos de MATLAB, sino ir más allá y poder crear un ejemplo que simule un modelo en Simulink y nos muestre los resultados tanto gráficamente como en una tabla. Una vez conseguido esto, se puede complicar el ejemplo consiguiendo modificar más parámetros del modelo a simular. Incluso se podría ir más allá y crear un ejemplo que se conecte a una planta real, se realice el experimento y que devuelva los datos gráficamente o en una tabla, todo esto remotamente. En los apartados siguientes se detalla cómo se han ido consiguiendo estos objetivos.

3.2.1 Modelo Simulink sencillo

El primer paso es conseguir ejecutar un modelo Simulink sencillo remotamente. Con el modelo de la Figura 3.9 se pretende observar la respuesta de una función de transferencia sencilla frente a una entrada escalón. A continuación se detalla paso a paso cómo conseguir simular ese modelo de Simulink remotamente y que devuelva los datos en una tabla y gráficamente:

1) En la ruta C:/MATLAB7/toolbox/webserver/wsdemos, abrir el archivo matweb.conf con el bloc de notas o WordPad y añadir después de la línea en la que pone mlport 9999 el siguiente código:

   [websim1]

   mlserver=localhost

   mldir=C:/MATLAB7/toolbox/webserver/wsdemos/

   [websim2]

   mlserver=localhost

   mldir=C:/MATLAB7/toolbox/webserver/wsdemos/

   Con esto se consigue que MATLAB Web Server tenga información de que existen unos archivos llamados websim1.m y websim2.m, que son los archivos que se ejecutarán remotamente.

2) Es necesario crear diferentes archivos y guardarlos en la ruta C:/MATLAB7/toolbox/webserver/wsdemos. Los archivos que hay que crear son:

   - **modelo_simulink.mdl**

     Desde MATLAB es necesario acceder a Simulink y crear el modelo que se visualiza en la Figura 3.9 guardándolo con el nombre indicado.

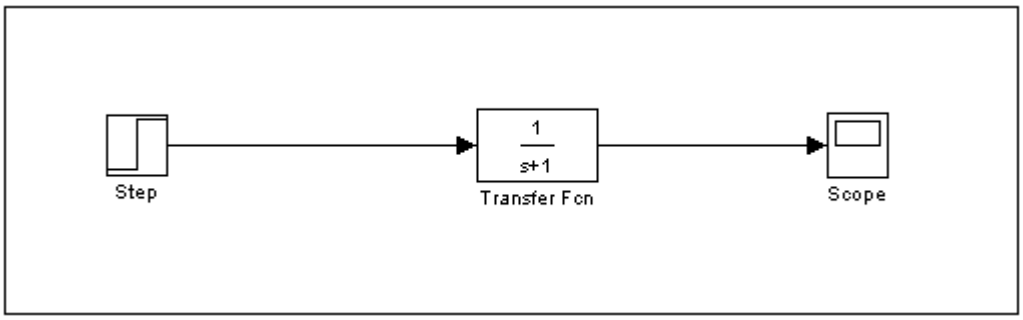


Figura 3.9. Modelo de Simulink sencillo

- **websim1.m** y **websim2.m**

Son los ficheros de MATLAB que se ejecutarán remotamente. Se crean mediante el editor de MATLAB. Cada archivo contiene la llamada a la simulación del modelo de Simulink modelo_simulink.mdl. Esta simulación se realiza con la instrucción de MATLAB sim('modelo_simulink', tiempo); en la que se puede indicar el tiempo de la simulación. El código completo de websim2.m comentado se encuentra en el anexo como ejemplo de todos los archivos de este tipo. No obstante, a continuación se detallan las líneas de código que realizan el cálculo de los resultados:

websim1.m

```
% Get the tiempo (string) variable. Convert to a number.
if(~length(instruct.tiempo))
  tiempo = 10; % Default empty field.
else
  tiempo = str2double(instruct.tiempo); % Extracción del parámetro tiempo
end

[t y]=sim('modelo_simulink', tiempo);   % Ejecución del modelo de Simulink

outstruct.t=t';
outstruct.y=y';

% Creación de la tabla con todos los datos
long=length(t);
tabla=zeros(long,2);
i=1;
```

```
while long>=i
   tabla(i,1)=t(i);
   i=i+1;
end
i=1;
while long>=i
   tabla(i,2)=y(i);
   i=i+1;
end
outstruct.tabla=tabla; % Creación de la tabla con los datos
```

websim2.m

```
% Get the tiempo (string) variable. Convert to a number.
if(~length(instruct.tiempo))
  tiempo = 10; % Default empty field.
else
  tiempo = str2double(instruct.tiempo); % Extracción del parámetro tiempo
end

load_system('modelo_simulink');
sim('modelo_simulink', tiempo); % Ejecución del modelo de simulink

t=ScopeData(:,1);
y=ScopeData(:,2);

% Creación de la gráfica
Fig = figure('visible','off'); % setup to create a jpeg file

plot(t,y);  % Gráfica con los datos
ylabel('Salida');
xlabel('Tiempo');
```

- **websim1a.html** y **websim2a.html**

Son dos ficheros HTML en los que se puede introducir el límite de tiempo de la simulación y tras pulsar "Ejecutar" lanzará el fichero de MATLAB que hará los cálculos. Se crean abriendo un bloc de notas o WordPad y guardándolo con extensión .html. El código completo de websim2a.html comentado se encuentra en el anexo como ejemplo de todos los archivos de este tipo. No obstante, a continuación se detallan las líneas de código más importantes:

websim1a.html

```
<p><font color="#000000" size="4" face="Arial">
<i>Ejecuta el siguiente modelo de simulink al pulsar Ejecutar</i></font>
</p>

<p><font color="#000000" size="4" face="Arial"><i><img
src="/icons/modelo_simulink.gif" width="512" height="158">
</i></font></p>

<form action="/cgi-bin/matweb.exe" method="POST">
   <input type="hidden" name="mlmfile" value="websim1">

   <p>Límite de tiempo:
   <input type="text" size="2" maxlength="3" name="tiempo"></p>

   <p><input type="submit" name="Ejecutar" value="Ejecutar"></p>
</form>
```

websim2a.html

Tiene el mismo código de websim1a.html exceptuando que llama a websim2.m

- **websim1b.html** y **websim2b.html**

En el fichero websim1b.html se puede ver una tabla con los datos que calculó la función websim1.m de la simulación. Con el fichero websim2b.html se puede ver una gráfica con los datos obtenidos de ejecutar la función websim2.m. Se crean del mismo modo que websim1a.html y websim2a.html. El código completo de websim2b.html comentado se encuentra en el anexo como

ejemplo de todos los archivos de este tipo. No obstante, a continuación se detallan las líneas de código más importantes:

websim1b.html

```
<div align="center">
<strong>Valores de la simulación obtenida de Simulink</strong></p>

<table border="1" cellspacing="1" cellpadding="4" autogenerate="$tabla$">
   <tr>
      <th>Tiempo
      <th>Salida
   </tr>
   <tr>
      <td align="center">
    </td>
   </tr>
</table>
<br>
</div>
```

websim2b.html

```
<div align="center">
<strong>Gráfica de la simulación obtenida de Simulink</strong></p>

<p align="center">
<img border=0 src="$GraphFileName$">
</p>
```

- **index.html**

Este fichero ya está creado, pero es necesario modificarle introduciendo la entrada a los ejemplos creados. Es un fichero HTML en el que se muestran las opciones para elegir la aplicación que se quiere usar. En este caso nos da dos opciones, una que mostrará los resultados en una tabla y la otra que los mostrará en una gráfica. El código completo de index.html comentado se encuentra en el anexo. No obstante, a continuación se detallan las líneas de código más

importantes que incluye los cuatro ejemplos básicos (de los dos últimos se hablará a continuación):

```
<p><font color="#000000" size="4" face="Arial"><i><img
src="/icons/logo_uc3m.gif" width="100" height="100"></i></font> <font
size="5"><strong><b>MATLAB Web Server Demo</b></strong></font> </p>

<ul>
  <li><a href="websim1a.html">Ejecución de modelo_simulink.mdl (websim1a)</a> Ejecuta un fichero de simulink y devuelve los datos de la simulación.<font face="Arial Narrow"></font> Los resultados se muestran automáticamente en una tabla mediante MATLAB Web Server.</li>
</ul>

<ul>
  <li><a href="websim2a.html">Ejecución de modelo_simulink.mdl (websim2a)</a> Ejecuta un fichero de simulink y devuelve los datos de la simulación.<font face="Arial Narrow"></font> Los resultados se muestran automáticamente en una gráfica mediante MATLAB Web Server.</li>
</ul>

<ul>
  <li><a href="websim3a.html">Ejecución de modelo_simulink_2.mdl (websim3a)</a> Ejecuta un fichero de simulink y devuelve los datos de la simulación. Se pueden establecer los valores del numerador y denominador de la función de transferencia.<font face="Arial Narrow"></font> Los resultados se muestran automáticamente en una gráfica mediante MATLAB Web Server.</li>
</ul>

<ul>
  <li><a href="websim4a.html">Ejecución del motor (websim4a)</a> Ejecuta un fichero de MATLAB que se conecta con el motor y devuelve los datos del ensayo.<Font face="Arial Narrow"></font> Los resultados se muestran automáticamente en una gráfica mediante MATLAB Web Server.</li>
</ul>
```

3.2.2 Modelo Simulink de mayor complejidad

Una vez conseguida la simulación de un modelo de Simulink sencillo, se pretende simular un modelo de mayor complejidad en el que se puedan introducir los datos de la función de transferencia. Para ello es necesario crear un nuevo modelo de Simulink al que se le puedan variar los parámetros de la función de transferencia. Este modelo nuevo es modelo_simulink_2.mdl. Además, ya no es suficiente con la instrucción sim(), por lo que hay que usar la instrucción set_param().

Con el modelo de la Figura 3.10 se pretende observar la respuesta de una función de transferencia, a la que se puede indicar el numerador y denominador, frente a una entrada escalón. A continuación se detalla paso a paso cómo conseguir simular ese modelo de Simulink remotamente y que devuelva los datos gráficamente (ya que es mucho más didáctico poder ver los datos en una gráfica y no en una tabla):

1) En el archivo matweb.conf es necesario incluir el nombre de la nueva función:

[websim3]

mlserver=localhost

mldir=C:/MATLAB7/toolbox/webserver/wsdemos/

2) Es necesario crear los siguientes archivos:

- **modelo_simulink_2.mdl**

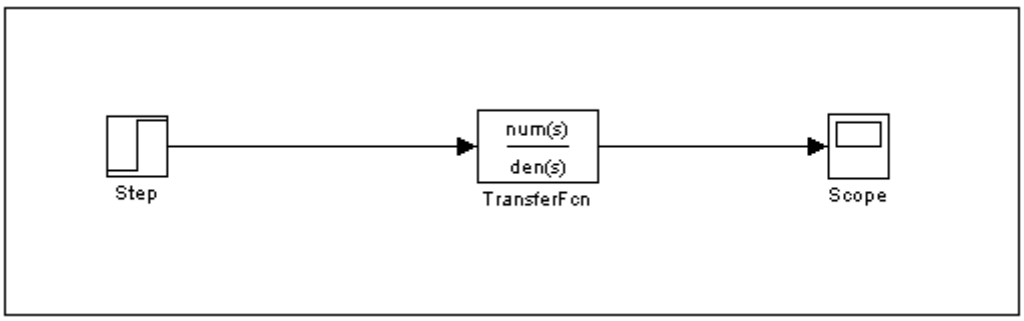


Figura 3.10. Modelo de Simulink de mayor complejidad

- **websim3.m**

En este fichero es necesario extraer los parámetros que se pasaron vía HTML y crear un array con los parámetros del denominador. Después de esto hay que convertir esas variables con la función mat2str(), e introducirlas en el modelo de

Simulink con la función set_param(). Finalmente el modelo se ejecuta con la instrucción sim(). Las líneas de código más importantes son:

```
% Get the tiempo (string) variable. Convert to a number.
if(~length(instruct.tiempo))
  tiempo = 10; % Default empty field.
else
  tiempo = str2double(instruct.tiempo); % Extracción del parámetro tiempo
end
if(~length(instruct.num))
  num = 1; % Default empty field.
else
  num = str2double(instruct.num); % Extracción del parámetro numerador
end
if(~length(instruct.den0))
  den0 = 1; % Default empty field.
else
  den0 = str2double(instruct.den0); % Extracción del primer parámetro del denominador
end
if(~length(instruct.den1))
  den1 = 1; % Default empty field.
else
  den1 = str2double(instruct.den1); % Extracción del segundo parámetro del denominador
end

den=[den0 den1];

load_system('modelo_simulink_2');
set_param('modelo_simulink_2/TransferFcn','Numerator',mat2str(num),'Denominator',mat2str(den)); % Introducción de los valores del numerador y denominador en la función de transferencia
```

sim('modelo_simulink_2', tiempo); % Ejecución del modelo de simulink

- **websim3a.html** y **websim3b.html**

  Ficheros HTML creados de manera similar a websim1a.html y websim1b.html.

  websim3a.html

  Difiere de wensim2a.html en que tiene que recoger más parámetros. Esto se consigue con las siguientes líneas de código:

```
<p>Tiempo:
<input type="text" size="5" maxlength="10" name="tiempo"></p>
<p>Numerador:
<input type="text" size="5" maxlength="10" name="num"></p>
<p>Denominador:
<input type="text" size="5" maxlength="10" name="den0">
<input type="text" size="5" maxlength="10" name="den1"></p>
```

  websim3b.html

  Igual que websim2b.html.

- **index.html**

  Es preciso modificar este fichero para incluir la opción de ejecutar el modelo de Simulink de mayor complejidad.

El ejemplo creado tiene un término en el numerador y dos términos en el denominador. Del mismo modo se puede proceder a crear un ejemplo con más términos, tanto en el numerador como en el denominador. Incluso se puede crear un ejemplo con más bloques de Simulink a los que se puede acceder del mismo modo con la función set_param() e ir cambiando los valores de los bloques. Con el fin de que los alumnos puedan aprender, se puede crear un modelo de Simulink de mucha más complejidad.

3.3 Conexión con Simulink en modo normal

Los modelos simulados hasta ahora han sido modelos de Simulink en modo normal. En este modo MATLAB Web Server es capaz de conectarse remotamente con el modelo de Simulink sin ningún tipo de problema. En la Figura 3.11 se puede observar cómo es el modo normal en Simulink.

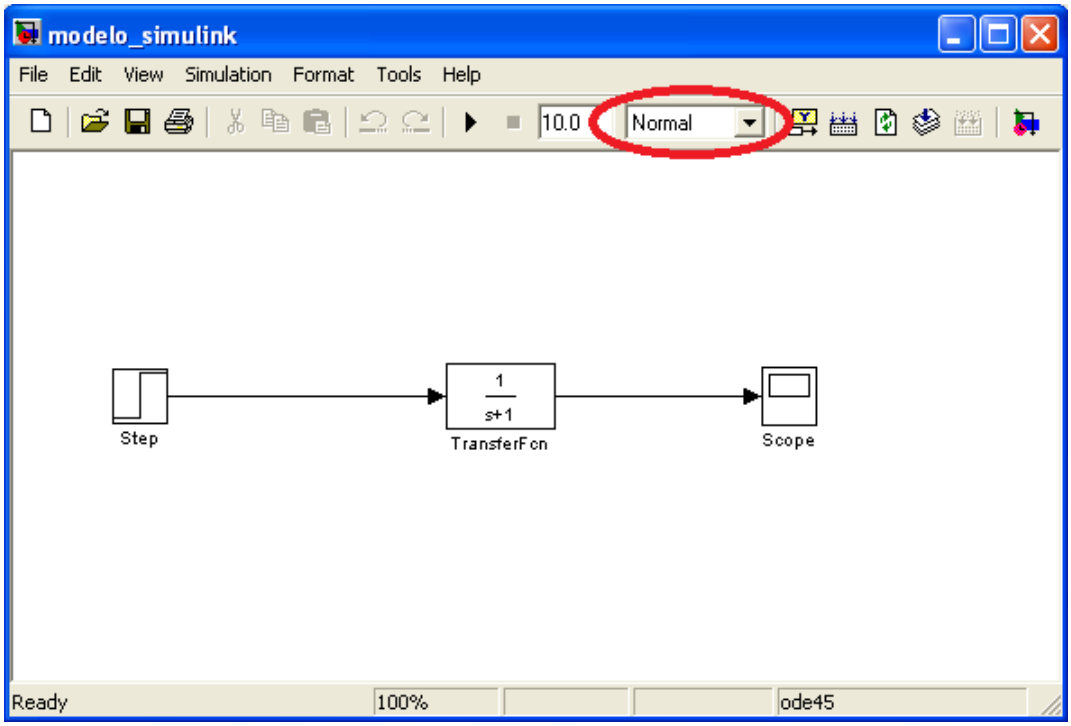


Figura 3.11. Modelo de Simulink en modo normal

3.4 Conexión con Simulink en modo externo

Existe un aspecto muy interesante de cara a realizar prácticas remotas. Este aspecto es poder conectarse con la maqueta del laboratorio, la cual tiene un motor, de forma remota. En este proyecto se ha perseguido ese fin, pero finalmente no se pudo conseguir del modo que se tenía pensado en un principio. No obstante se consiguió de otro modo que se detalle en el siguiente punto.

El modo que se quería usar en un principio era poder conectarse remotamente con un modelo Simulink que tuviera entrada y salida analógicas como el de la Figura 3.12 para poder conectar con el motor real en modo externo como se indica en la figura.

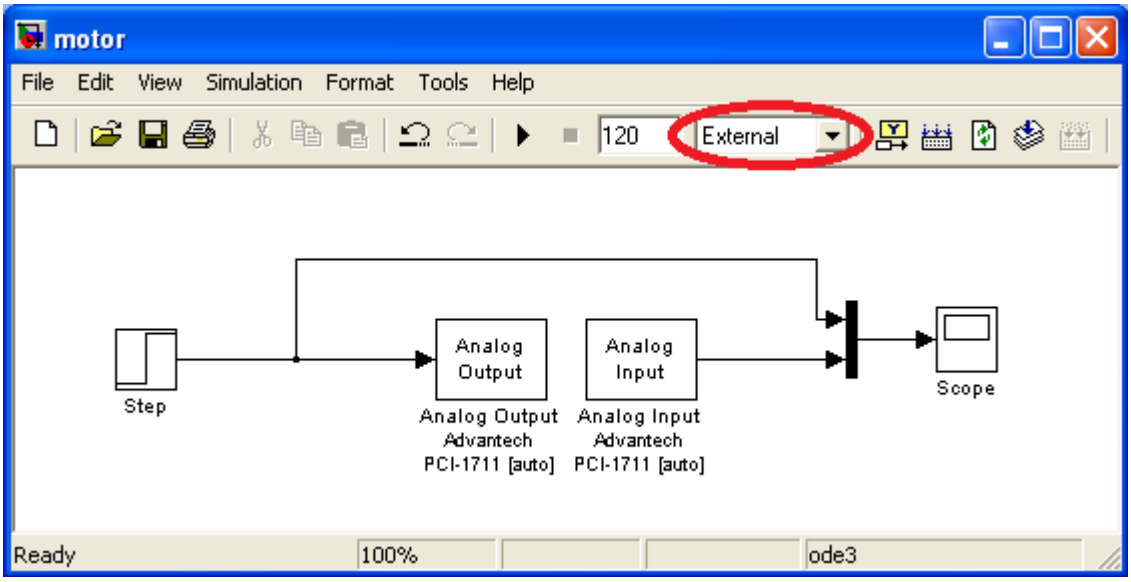


Figura 3.12. Modelo de Simulink con entrada y salida analógicas en modo externo

Después de varios intentos, búsqueda de información con el fin de solucionar el problema, contacto con profesores que habían usado esta tecnología y muchas horas de trabajo, se llegó a la conclusión de que MATLAB Web Server no funciona con Simulink en modo externo. Esta conclusión está apoyada por el artículo "Acceso remoto a ensayos de control en tiempo real basados en MATLAB" (Rafael Palacios, Ramón Rodríguez Pecharromán, 2005).

Se realizaron pruebas de forma local, una vez que funcionaba de forma local se intentaba de forma remota sin éxito. Las diferentes opciones que se barajaron para intentar hacer funcionar MATLAB Web Server en modo externo se resumen a continuación:

1) El primer intento se hizo de forma local ejecutando un script con los siguientes comandos:

```
load_system('modelo_simulink');
set_param('modelo_simulink','SimulationMode','external');
set_param('modelo_simulink','SimulationCommand','connect');
set_param('modelo_simulink','StopTime','5');
set_param('modelo_simulink','SimulationCommand','start');
set_param('modelo_simulink_2','SimulationCommand','stop');
set_param('modelo_simulink_2','SimulationCommand','disconnect');
t=ScopeData(:,1);
y=ScopeData(:,2);
plot(t,y);
```

No funciona porque la variable ScopeData no está accesible hasta que no finaliza la simulación.

2) El segundo intento se realizó a partir del primero, pero programando un timer que detuviera la simulación. Esto funcionaba de forma local pero no de forma remota.

3) Otro intento fue realizando dentro de un bucle infinito una espera al fin de la simulación para imprimir los datos. La espera se realizaba con el siguiente código:

```
while(1)
   time=get_param('modelo_simulink','SimulationTime');
   %pause(1)
   if(time>0)
      t=true;
   elseif(t==true)
      break
   end
end
```

Tampoco funcionó, se quedaba bloqueado esperando en el while().

4) El siguiente intento fue llamar a un script que empezara la simulación y después de transcurrido un tiempo llamar a un segundo script que detuviera la simulación. Esto funcionaba perfectamente en modo local, pero al pasarlo a modo remoto, intercalando una página HTML en la que había que pulsar un botón para detener la simulación, no funcionaba.
También se intentó que se llamara a un script dentro de otro y tampoco fue satisfactorio.

5) Otro intento fue realizando una espera para imprimir los datos, o incluso una espera para finalizar la simulación y otra para imprimir los datos. Se realizó con este código:

```
t1 = timer('TimerFcn', 'stat=false;','StartDelay',1);
start(t1)

stat=true;
while(stat==true)
  time=get_param('motor_pid_2','SimulationTime');
end

delete(t1)
```

Tampoco fue satisfactoria esta solución.

6) Otro intento dio una falsa esperanza, porque funcionó en remoto pero en modo normal y al ponerlo en modo externo no funcionó. El intento fue usar dos scripts, uno que arrancaba la simulación y otro que la paraba cuando se daba a un botón desde una página HTML intermedia entre la página HTML de entrada de datos y la página HTML de visualización de los resultados.

7) Otro intento fue usar los bloques ToWorkspace y ToFile, con el fin de recoger los datos pero en modo remoto no funcionaba, al igual que recogiendo los datos del ScopeData.

Éstos son los intentos más relevantes que se probaron, no obstante, hubo muchos intentos con pequeños cambios que no se reflejan aquí. Como por ejemplo probar con un modelo de Simulink sin motor, después con un modelo con entrada y salida analógica que conectaba con el motor, etc. Fueron muchas horas de laboratorio y de búsqueda de información intentando que funcionara de una manera u otra.

3.5 Conexión con el motor usando Data Acquisition Toolbox

Después de todos las pruebas intentando que el sistema funcionara usando Simulink en modo externo, se empezó a pensar en no usar Simulink e intentar acceder al motor desde MATLAB. Se sabía que MATLAB Web Server era capaz de conectarse con el PC remoto y ejecutar un fichero de MATLAB. Sólo se necesitaban unas instrucciones que permitieran conectarse directamente al motor, porque el fichero .m se leía perfectamente.

De ahí salió la idea de usar la Data Acquisition Toolbox de MATLAB.

A continuación se detalla paso a paso cómo se consiguió conectarse al motor remotamente:

1) En el archivo matweb.conf es necesario incluir el nombre de la nueva función:

[websim4]

mlserver=localhost

mldir=C:/MATLAB7/toolbox/webserver/wsdemos/

2) Es necesario crear los siguientes archivos:

- **websim4.m**

En este fichero están las instrucciones de MATLAB que usan la Data Acquisition Toolbox que permiten conectarse con el motor. A continuación se detallan las instrucciones con comentarios de qué es lo que hace cada una:

```
ao = analogoutput('advantech'); % Indica la tarjeta de adquisición de datos
addchannel(ao,1:1); % Especifica el canal por el que enviar información
set(ao,'SampleRate',500) % Especifica la frecuencia de muestreo
putdata(ao,7) % Especifica la tensión a la salida
start(ao) % Arranca el envío de datos
ai = analoginput('advantech'); % Indica la tarjeta de adquisición de datos
addchannel(ai,1:1); % Especifica el canal por el que recibir información
set(ai,'SampleRate',500) % Especifica la frecuencia de muestreo
set(ai,'SamplesPerTrigger',tiempo*500) % 500 muestras en un segundo
start(ai) % Arranca el recibimiento de datos
data = getdata(ai); % Almacena los datos recibidos en la variable data
t = [0:1/(500):tiempo-1/(500)]'; % Crea un vector para representar la entrada
entrada(1)=0;
entrada(2:500*tiempo)=7;

putdata(ao,2) % Cambia la tensión de salida
start(ao)
delete(ao); % Elimina de memoria para liberar espacio
delete(ai);
clear ao
clear ai
```

- **websim4a.html** y **websim4b.html**

Ficheros HTML creados de manera similar a websim1a.html y websim1b.html.

- **index.html**

  Es preciso modificar este fichero para incluir la opción de ejecutar el fichero de MATLAB que se conecta con el motor y devuelve los datos del ensayo.

De este modo se consiguió la conexión con el motor que tanto se buscó conseguir con Simulink en modo externo sin éxito.

Capítulo 4

Solución con Easy Java Simulations

4 Solución con Easy Java Simulations

Easy Java Simulations, también conocido como EJS, es una herramienta de autor creada en Java que ayuda a personas que no tienen conocimientos de programación a crear simulaciones interactivas en Java, habitualmente con fines de enseñanza o aprendizaje. El fin didáctico es el que se busca en este proyecto, debido a esto se ha decidido usar esta herramienta. Es una herramienta muy intuitiva y permite la conexión con MATLAB y Simulink, aspecto que interesa en este proyecto. Además, permite realizar estas conexiones de forma remota, por lo que se conseguiría tener un laboratorio virtual y/o remoto.

EJS ha sido creado por Francisco Esquembre y es parte del proyecto Open Source Physics (Física de código abierto).

Primeramente, se va a explicar qué es Easy Java Simulations con más detalle. Más tarde, se explicará cómo usar esta herramienta y cómo crear ejemplos conectados a MATLAB y Simulink.

Easy Java Simulations es una herramienta de software diseñada para la creación de simulaciones discretas por computador. Una simulación discreta por computador, o simplemente una simulación por computador, es un programa de computador que intenta reproducir, ya sea con fines pedagógicos o científicos, un fenómeno natural a través de la visualización de los diferentes estados que éste puede presentar. Cada uno de estos estados está descrito por un conjunto de variables que cambia en el tiempo debido a la iteración de un cierto algoritmo.

En la Figura 4.1, se puede observar un ejemplo de una simulación en EJS. Se trata de una masa y un muelle. En la parte de la derecha de la figura se visualizará la evolución del desplazamiento y velocidad con respecto al tiempo.

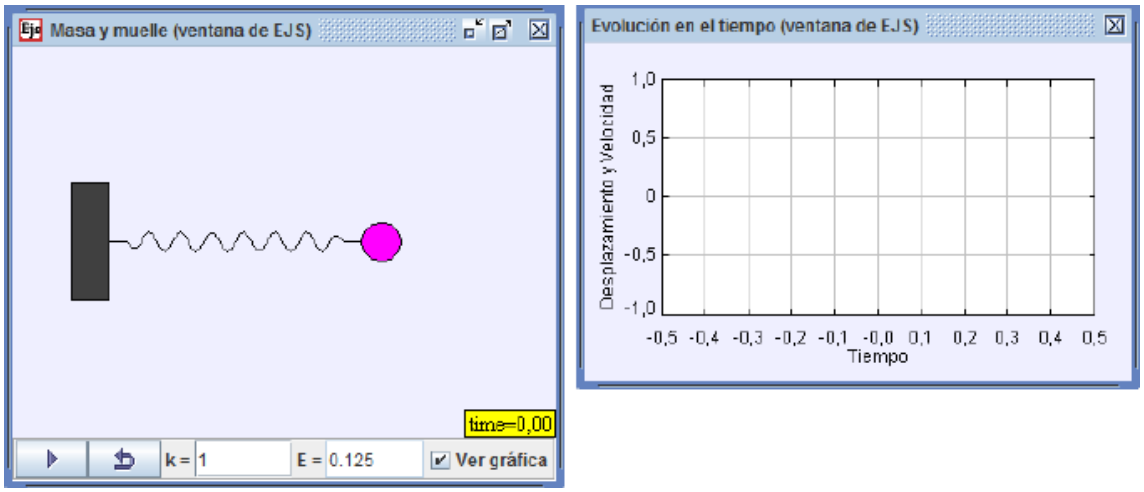


Figura 4.1. Ejemplo de masa y muelle en EJS

Para conseguir que haya un movimiento de la masa, es necesario definir la ecuación del fenómeno físico que se tiene. Con esta información, EJS puede simular el fenómeno físico como si fuese un fenómeno natural, a través de la visualización de los diferentes estados por los que pasa el fenómeno natural. Antes de definir las ecuaciones es necesario definir las variables que se usarán como se muestra en la Figura 4.2.

Figura 4.2. Definición de las variables en EJS

En la Figura 4.3 se muestran las ecuaciones que se corresponden con el fenómeno físico de una masa y un muelle.

Figura 4.3. Definición de las ecuaciones en EJS

También es necesario determinar la Evolución o las Relaciones fijas. En este caso se ha usado el apartado Relaciones físicas porque se cumplen siempre, pero en los casos de conexión a Simulink por ejemplo, se usará el apartado Evolución. En la Figura 4.4 se muestra la definición de las relaciones fijas.

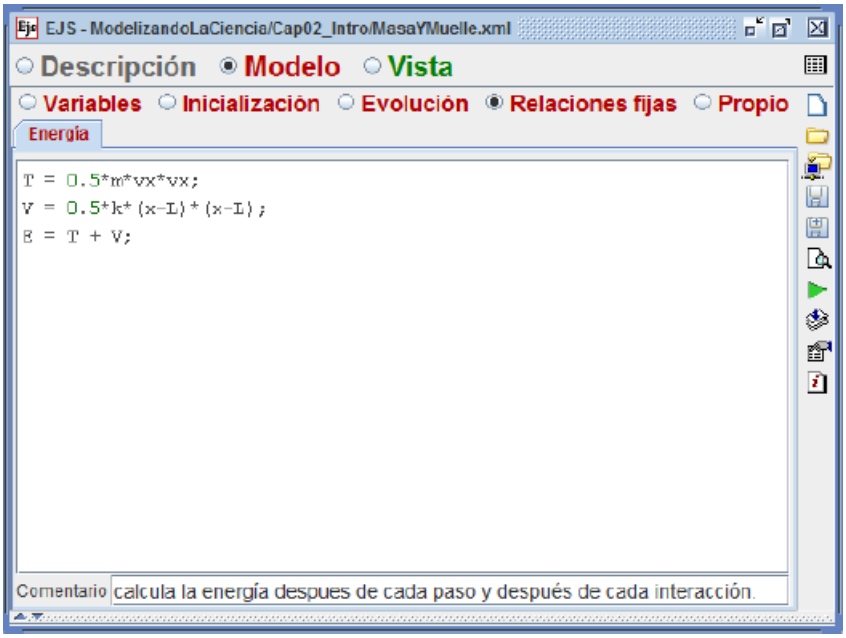


Figura 4.4. Definición de las relaciones fijas en EJS

Con todo lo visto anteriormente, se puede decir que EJS es un programa que ayuda a crear simulaciones científicas.

Existen muchos programas que ayudan a crear otros programas, pero EJS se diferencia de la mayoría de los otros programas porque no ha sido diseñado para hacer la vida más sencilla a los programadores profesionales, sino que ha sido concebido por profesores de ciencias para ser usado por profesores y estudiantes de ciencias. Con esto se consigue que personas sin conocimientos de programación, que sólo les interesa el fenómeno físico que se simula, sean capaces de crear su simulación, sin tener en cuenta los aspectos técnicos necesarios para construir la simulación.

Easy Java Simulations es una herramienta de modelado expresamente dedicada a esta tarea. Ha sido diseñado para permitir a sus usuarios (en su mayoría sin conocimientos de programación) trabajar a un alto nivel conceptual, usando un conjunto de herramientas simplificadas y concentrando la importancia en los aspectos científicos de la simulación. Se ha conseguido que el computador realice automáticamente todas las otras tareas necesarias que son fácilmente automatizables.

No obstante lo anterior, el resultado final, generado automáticamente por EJS a partir de la descripción del autor, podría pasar, en términos de eficiencia y sofisticación, por la creación de un programador profesional. De ahí viene la importancia de esta herramienta, que en un futuro puede servir para múltiples opciones.

En particular, EJS crea aplicaciones Java que son independientes y multiplataforma, o applets que se pueden visualizar usando cualquier navegador Web (y por tanto ser distribuidos a través de Internet), que pueden leer datos a través de la red y ser controlados usando scripts (conjuntos de instrucciones) incluidos en las páginas HTML. Por estos motivos es una idea muy potente que puede dar salida a soluciones muy diferentes.

EJS puede ser usado también como una herramienta pedagógica, ya que las simulaciones tienen un valor educativo bastante importante. Es mucho más fácil la enseñanza de fenómenos físicos mediante simulaciones que son muy intuitivas. Los profesores pueden pedir a sus estudiantes que creen una simulación por sí mismos usando EJS que les puede ayudar a que hagan explícitos sus conocimientos y conceptos. Usado en grupos, EJS puede servir también para mejorar las capacidades de los alumnos para discutir y comunicarse sobre cuestiones científicas.

Con todo esto, se tiene una visión general de qué es EJS. En los siguientes apartados se va a explicar la parte de conexión de EJS con MATLAB y Simulink que es lo que nos interesa en este proyecto.

Para poder usar Easy Java Simulations de forma remota es necesario disponer de dos PCs mínimo. Uno será el servidor, que tendrá instalado un servidor, MATLAB con Simulink y dispondrá de conexión a internet. El otro PC deberá tener conexión a internet, un explorador de internet y Java instalados. En la Figura 4.5 se puede observar los requisitos del sistema esquemáticamente.

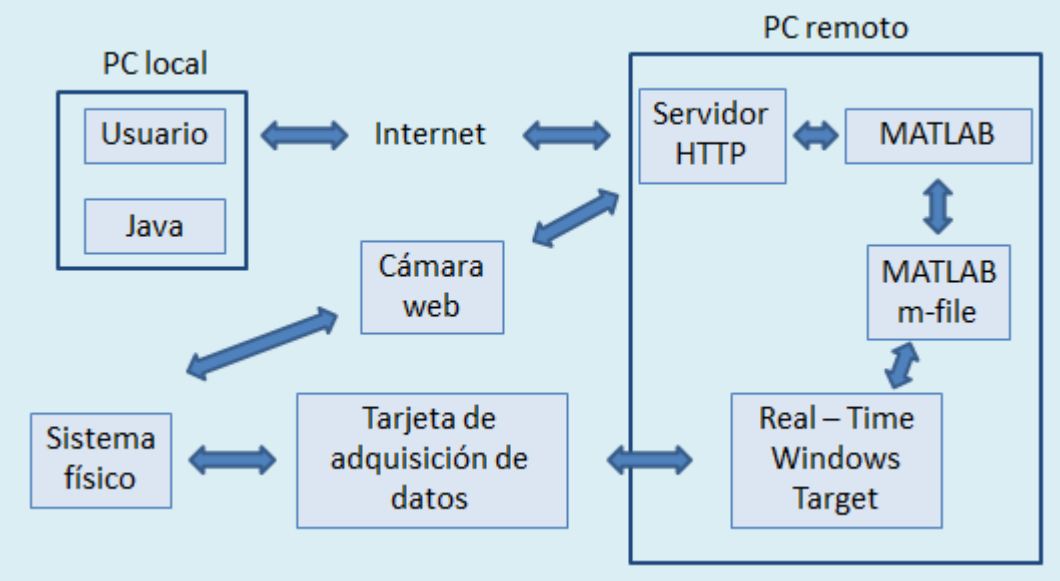


Figura 4.5. Esquema detallado del funcionamiento del sistema con EJS

Una vez se tienen los ejemplos funcionando de forma local, es muy sencillo incluirlos en la página web http://docenciaisa.uc3m.es/ para poder tener acceso remoto a ellos. De este modo, los alumnos podrán realizar prácticas de forma remota.

4.1 Conexión con Simulink en modo normal

Se ha creado un ejemplo muy sencillo que puede conectarse con Simulink en modo normal. Este ejemplo tiene una interfaz gráfica sencilla, con una gráfica en la que se van dibujando los datos según los recibe de Simulink, un botón de "Play" y otro de "Stop", para arrancar o parar la simulación respectivamente. En la Figura 4.6 se puede observar la interfaz gráfica.

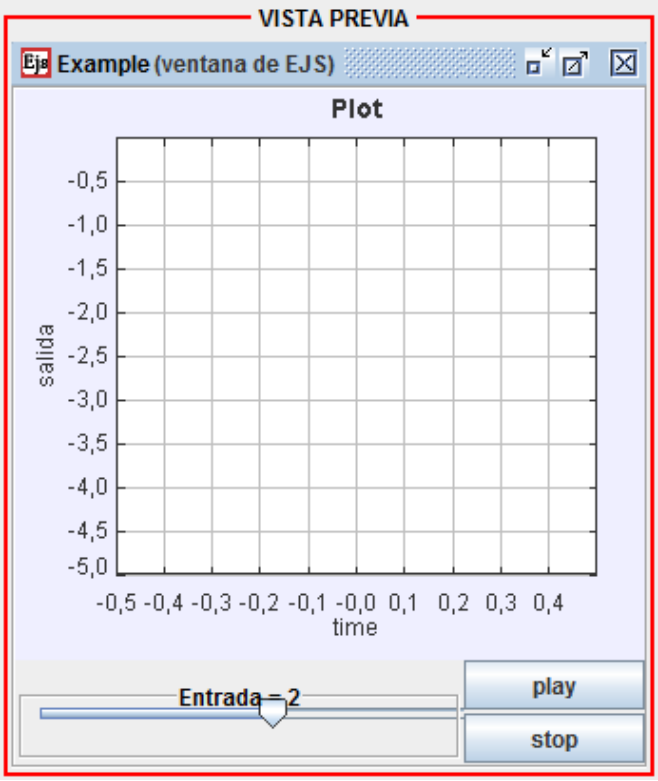


Figura 4.6. Interfaz gráfica

Para crear este ejemplo es preciso disponer de una instalación de MATLAB con Simulink, al igual que EJS. A continuación se detalla paso a paso cómo se debe de proceder para conseguir conectar EJS con Simulink:

1) Es necesario crear un modelo de Simulink que será el modelo que se simule. Se ha optado por crear un modelo sencillo, ya que el objetivo de este proyecto no es complicar los modelos, sino conseguir establecer la conexión. El modelo se puede observar en la Figura 4.7. Se va a estudiar la respuesta de una función de transferencia ante una entrada escalón.

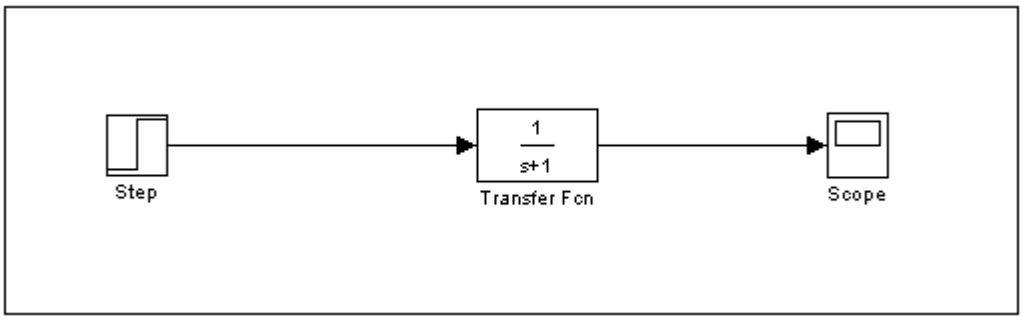


Figura 4.7. Modelo de Simulink sencillo

2) Una vez creado el modelo de Simulink hay que indicar en EJS que se va a usar este archivo como un archivo propio especificando la ruta donde se encuentra como se puede observar en la Figura 4.8.

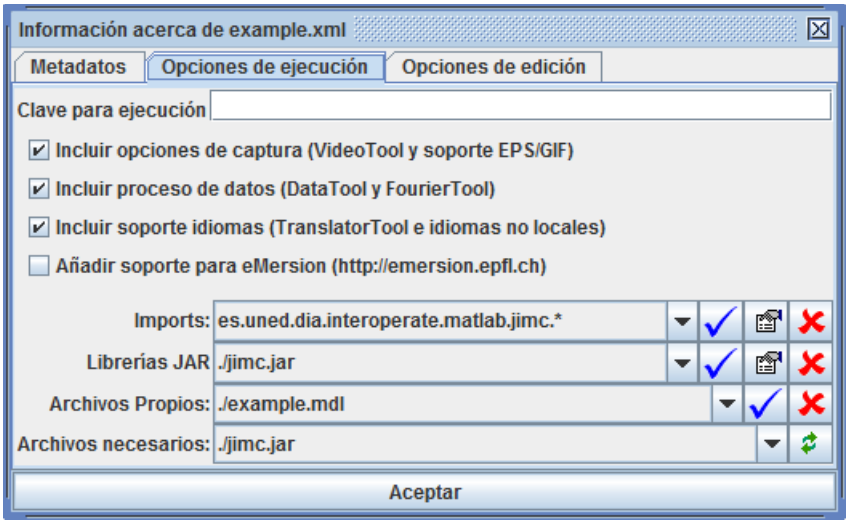


Figura 4.8. Ventana para indicar ruta del archivo propio

3) El siguiente paso es crear las variables de EJS. Esto se realiza en la pestaña Modelo/Variables como se muestra en la Figura 4.9.

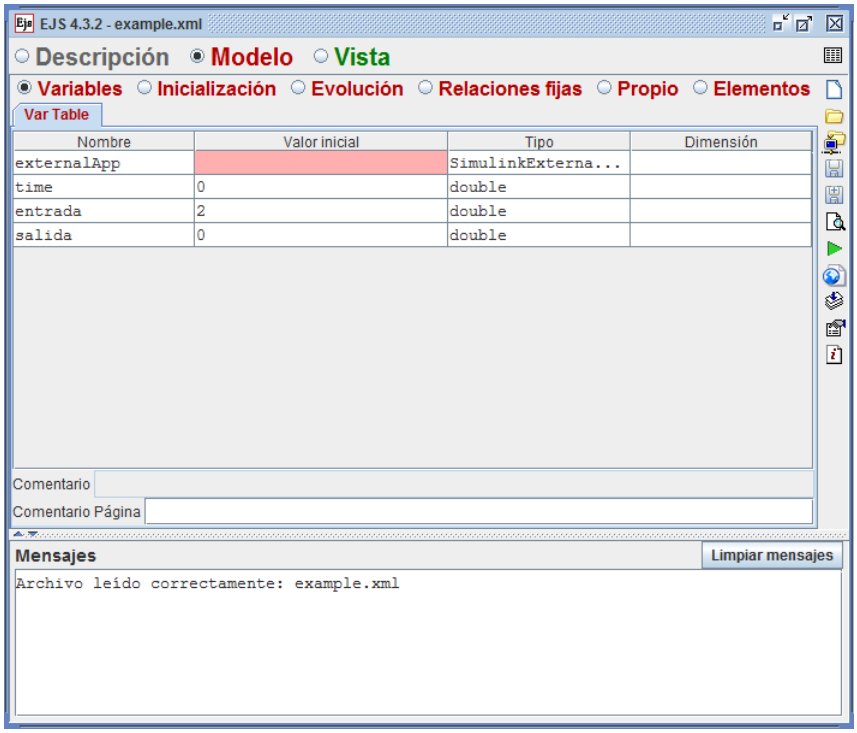


Figura 4.9. Ventana para indicar las variables

4) Después de esto es necesario vincular las variables de EJS con las de Simulink. Esto se realiza en la pestaña Modelo/Inicialización como se muestra en la Figura 4.10.

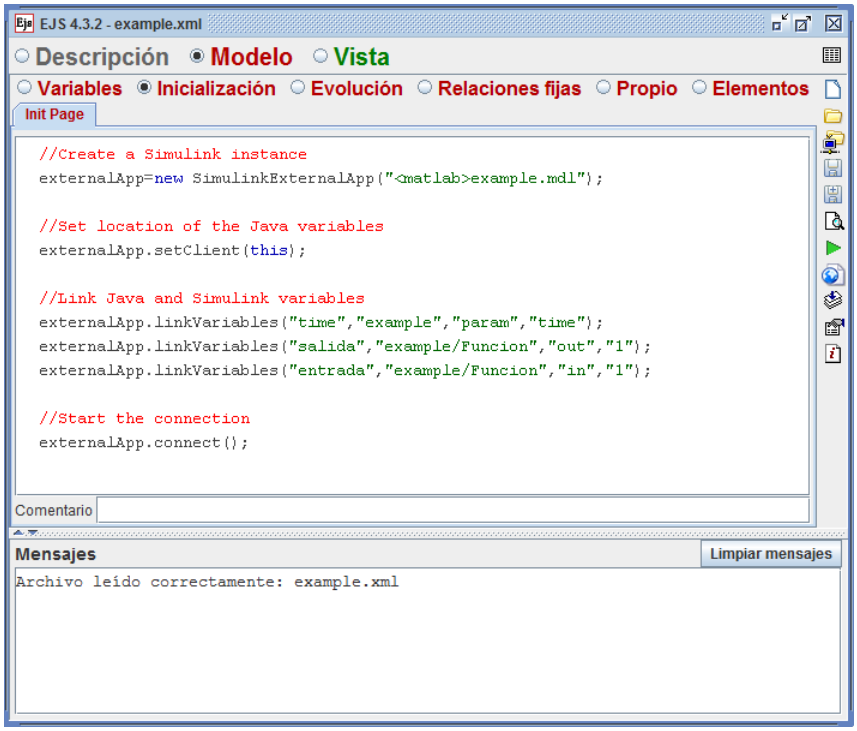


Figura 4.10. Ventana para vincular las variables con Simulink

5) En la pestaña Modelo/Evolución sólo es necesario incluir la siguiente línea de código: externalApp.step(1);

6) Por último en la pestaña Vista se diseña la interfaz gráfica como la de la Figura 4.6. Se ha creado una gráfica con dos trazas, una dibuja la entrada y la otra la salida. Esto hay que indicarlo en las características de las trazas, al igual que las acciones de los botones "Play" y "Stop". En la Figura 4.11 se muestra un ejemplo de cómo se indica la función del botón "Play". Los demás objetos de la interfaz gráfica se realizan de modo similar.

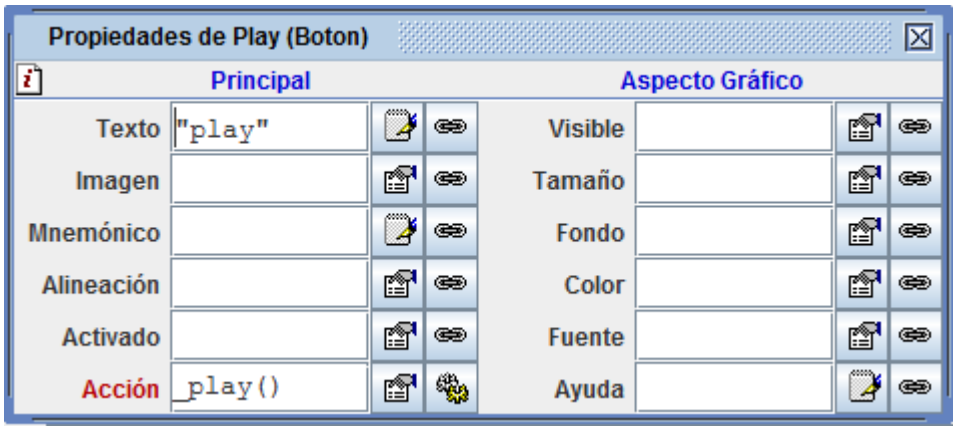


Figura 4.11. Propiedades del botón "Play"

Con esto se tiene diseñado el ejemplo para que funcione correctamente y se puedan recibir los datos de Simulink en la interfaz gráfica de EJS. En el capítulo Resultados experimentales se puede observar mediante imágenes cómo funciona.

4.2 Conexión con Simulink en modo externo

Al igual que se intentó usando la tecnología de MATLAB Web Server, también se ha intentado conectar con Simulink en modo externo usando la tecnología de Easy Java Simulations, con el fin de poder conectar remotamente con un motor, pero no ha sido posible.

El problema es que EJS tiene una limitación que hace imposible que se pueda conectar EJS con Simulink en modo externo. Los creadores de EJS tienen que depurar este aspecto para que esto pueda funcionar. Durante el desarrollo de este proyecto se ha contactado con ellos para intercambiar información y comunicar estos problemas. Durante una reunión con uno de los desarrolladores de EJS en la Universidad Carlos III de Madrid se concluyó que si el modelo compilase una vez solventados esos problemas funcionaría sin ningún problema.

El motivo por el que no funciona es porque EJS modifica el modelo .mdl de Simulink cambiando el modelo. En la Figura 3.12 se puede observar cómo EJS realiza cambios en el modelo de Simulink.

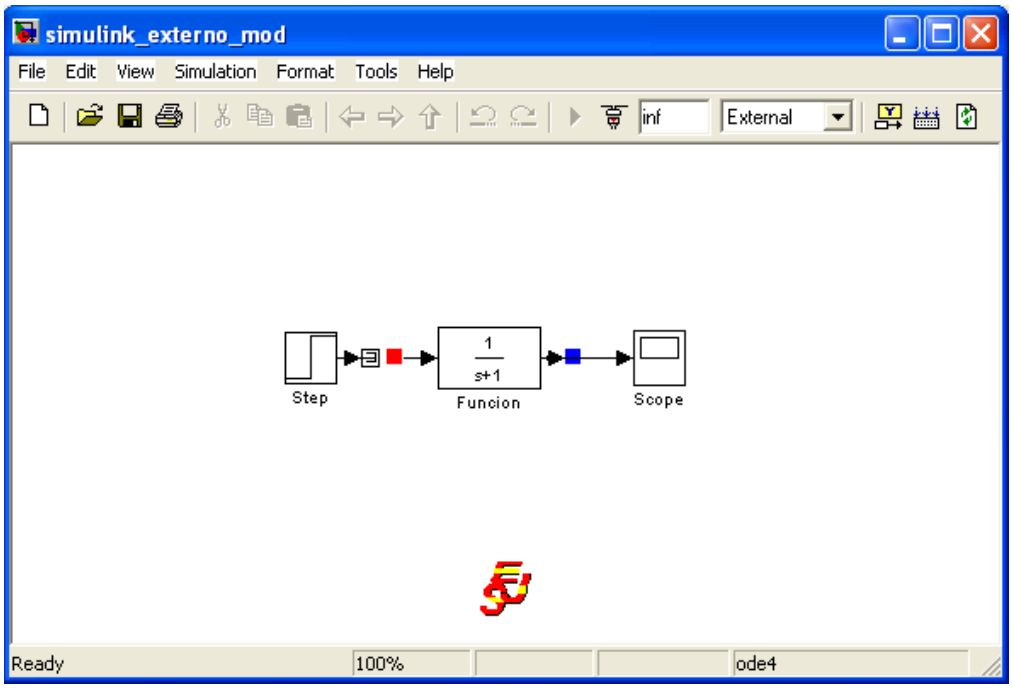


Figura 4.12. Modelo de Simulink modificado por EJS

Con otros modelos de Simulink, EJS modifica los bloques del modelo. Estos bloques modificados contienen unas funciones llamadas MATLAB function. No se tienen los TLC de estos bloques con MATLAB function, por lo tanto no pueden ser compilados. Sin poder compilar el modelo modificado por EJS nunca se conseguiría la conexión.

En el caso de que los creadores de EJS incluyeran los TLC de esos bloques o modificaran el modo en que se modifican los bloques de los modelos o los modelos, sí se podría conectar con el motor vía Simulink en modo externo, y del mismo modo remotamente.

4.3 Conexión con el motor usando Data Acquisition Toolbox

Otra vía estudiada para conectar Easy Java Simulations con el motor ha consistido en crear un modelo de Simulink con los bloques de "Analog Input" y "Analog Output" de la Data Acquisition toolbox. Ésta ha sido la manera de poder conectarse con el motor finalmente mediante Easy Java Simulations. Al comprobar que esto funcionaba con Easy Java Simulations se intentó con MATLAB Web Server, pero no fue posible ya que la versión de MATLAB que tiene MATLAB Web Server no tiene la Data Acquisition toolbox en Simulink, por lo tanto es inviable. En cambio sí es posible usando la Data Acquisition toolbox desde comandos de MATLAB. Este problema es común en los trabajos de investigación, se intenta solucionar algo por una vía que no siempre tiene por qué funcionar. En la Figura 4.13 se puede observar el modelo de Simulink usado.

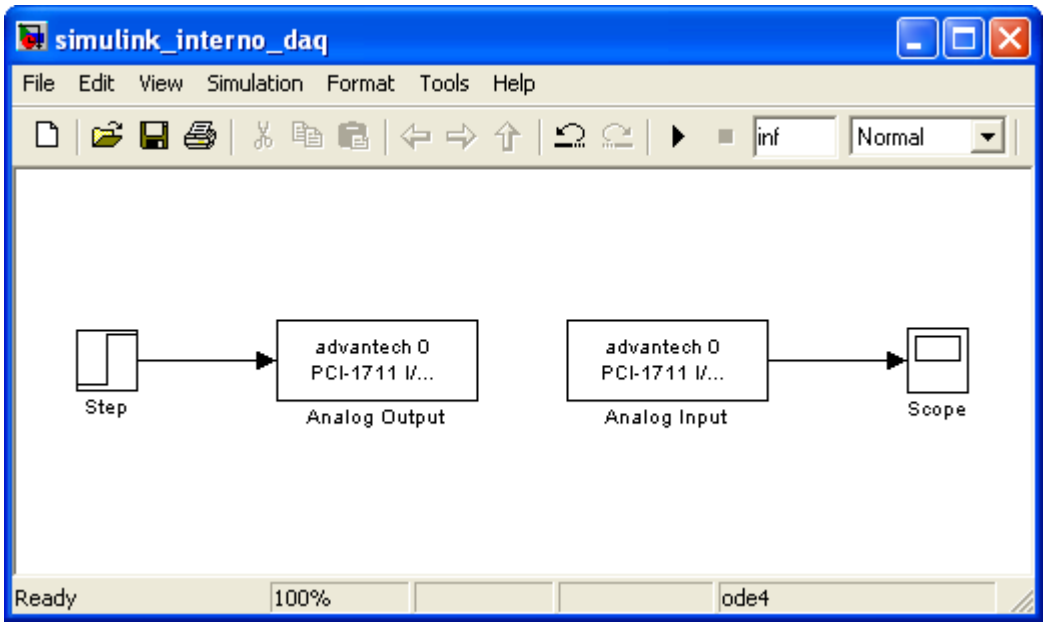


Figura 4.13. Modelo de Simulink con los bloques del Data Acquisition toolbox

Una vez conseguido con Simulink usando la Data Acquisition toolbox, se intentó mediante instrucciones de MATLAB que manejan la Data Acquisition toolbox, pero no se consiguió la conexión por problemas configurando la tarjeta de adquisición de datos. No obstante, si se solventan esos problemas de configuración, sería una tarea sencilla.

Capítulo 5

Resultados experimentales

5 Resultados experimentales

Para la validación de la investigación realizada se ha procedido a desarrollar las propuestas y probarlas con el equipo de laboratorio existente en el Departamento de Ingeniería de Sistemas y Automática de la Universidad Carlos III de Madrid.

A continuación se describen los equipos utilizados y se muestran las interfaces desarrolladas con imágenes de su funcionamiento utilizando MATLAB Web Server y Easy Java Simulations.

5.1 Equipo de laboratorio utilizado

Como se ha comentado anteriormente el sistema desarrollado en este proyecto se ha implementado y probado sobre el equipo de laboratorio existente. Este equipo consta de un PC el cual tiene conectado a uno de sus buses PCI la tarjeta de adquisición de datos "Advantech PCI-1711", que a su vez se conecta a una maqueta. En la parte superior de la Figura 5.1 se observa la tarjeta que va instalada en el PC, mientras que en la parte inferior se observa la tarjeta en la cual se realizan las conexiones físicas mediante cables con la maqueta.

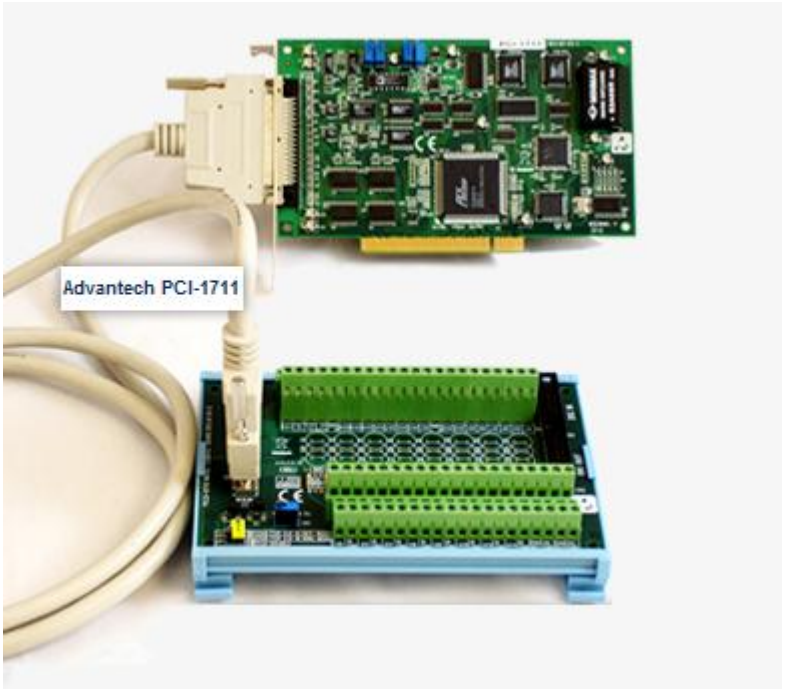


Figura 5.1. Tarjeta de adquisición de datos Advantech PCI-1711

Las características de esta tarjeta son las siguientes:

- Es una tarjeta PCI.
- Consta de 16 entradas digitales y 16 entradas digitales.
- Consta de 16 entradas analógicas.
- Consta de 2 salidas analógicas de 12 bits cada una.
- Tiene un convertidor A/D de 12 bits.

Estas características son más que suficientes para lo que se precisa en este proyecto, ya que el objetivo de este proyecto es la conexión de forma remota y para ello sólo son necesarias una salida y una entrada analógicas, además del convertidor A/D. La hoja de características de la tarjeta se encuentra adjuntada en el anexo para más información.

La maqueta consta de un motor de corriente continua controlado por armadura, con velocidad de 4000RPM a 24Vcc, que incorpora una tacogeneratriz como sensor de velocidad y un encoder como sensor de posición. En la Figura 5.2 se observa el motor de la maqueta.

Figura 5.2. Motor de la maqueta

Además de esto, la maqueta tiene la circuitería necesaria para poder conectarse con la tarjeta de adquisición de datos. En la parte superior de la maqueta hay un panel con conectores que permiten realizar las conexiones con la tarjeta. En la Figura 5.3 se pueden observar los conectores.

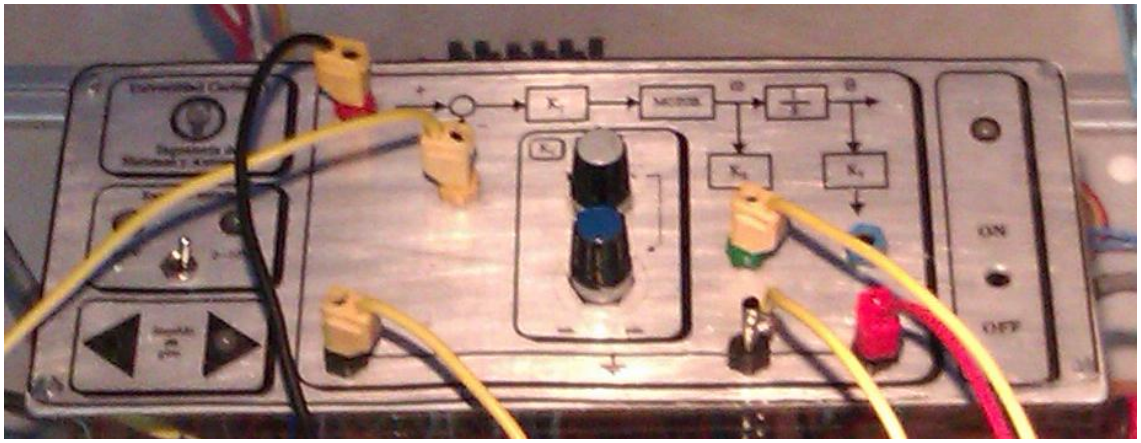


Figura 5.3. Conectores de la maqueta

Todo ello compone el sistema que se ha usado, en la Figura 5.4 se puede observar el sistema completo con el motor dentro de la maqueta conectado a la tarjeta de adquisición de datos.

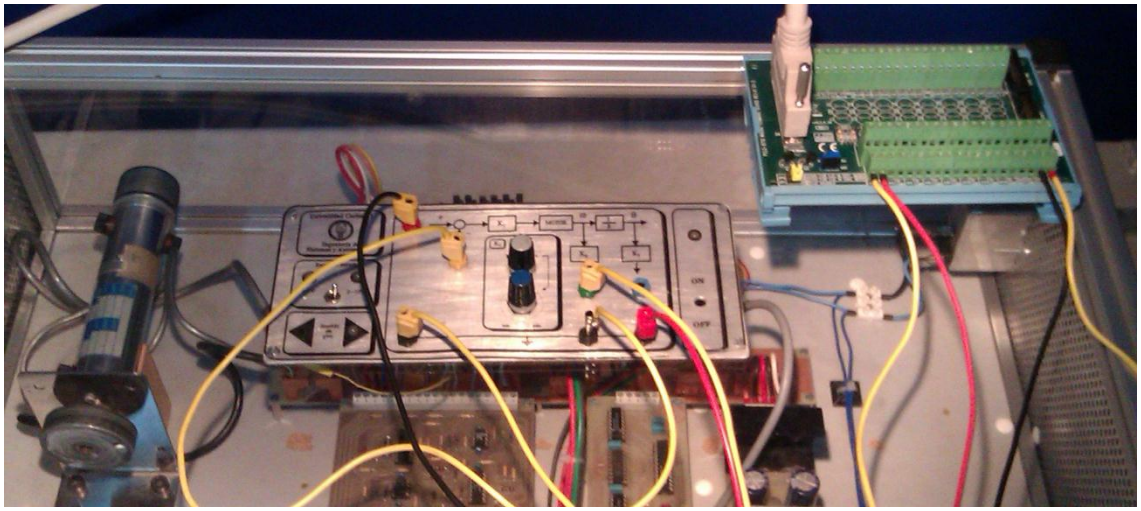


Figura 5.4. Sistema completo

5.2 Resultados

Para poder probar el funcionamiento de la conexión remotamente y poder visualizar los resultados se ha diseñado una interfaz gráfica mediante HTML. Es una interfaz muy básica ya que no es el objeto de este proyecto, el objeto de este proyecto es conseguir la conectividad.

Al escribir la IP pública del PC del laboratorio en la barra de direcciones de un navegador web o accediendo desde la página web http://docenciaisa.uc3m.es/ se puede elegir entre 5 opciones como se muestra en la Figura 5.5.

Figura 5.5. Página principal con las 5 opciones a elegir

Las cuatro primeras opciones son ejemplos desarrollados con MATLAB Web Server, mientras que la última opción permite la ejecución de dos ejemplos desarrollados con Easy Java Simulations.

El primer ejemplo de la página de la Figura 5.5 ejecuta un modelo de Simulink sencillo y devuelve los datos en una tabla. El segundo hace lo mismo, pero devolviendo los datos en una gráfica. El tercero hace lo mismo que el segundo pero se le puede indicar los parámetros de la función de transferencia. El cuarto ejecuta el motor y devuelve los resultados gráficamente. La última opción abre otra página web que ofrece la posibilidad de elegir entre dos ejemplos de Easy Java Simulations. El primer ejemplo ejecuta un modelo de Simulink sencillo y devuelve los datos gráficamente mientras se realiza la simulación. El segundo se conecta con el motor mediante Simulink y también devuelve los datos gráficamente mientras se realiza el ensayo. A continuación se detalla el funcionamiento de cada ejemplo en un apartado diferente.

5.2.1 Conexión mediante MATLAB Web Server con un modelo de Simulink sencillo con resultados mediante una tabla

Si se elige la primera opción, aparecerá otra página en la que se muestra el modelo a simular como se puede observar en la Figura 5.6. En esta página se puede especificar el tiempo total de la simulación. Al pulsar el botón "Ejecutar" se ejecutará un modelo de Simulink que devolverá los resultados en una tabla como se aprecia en la Figura 5.7.

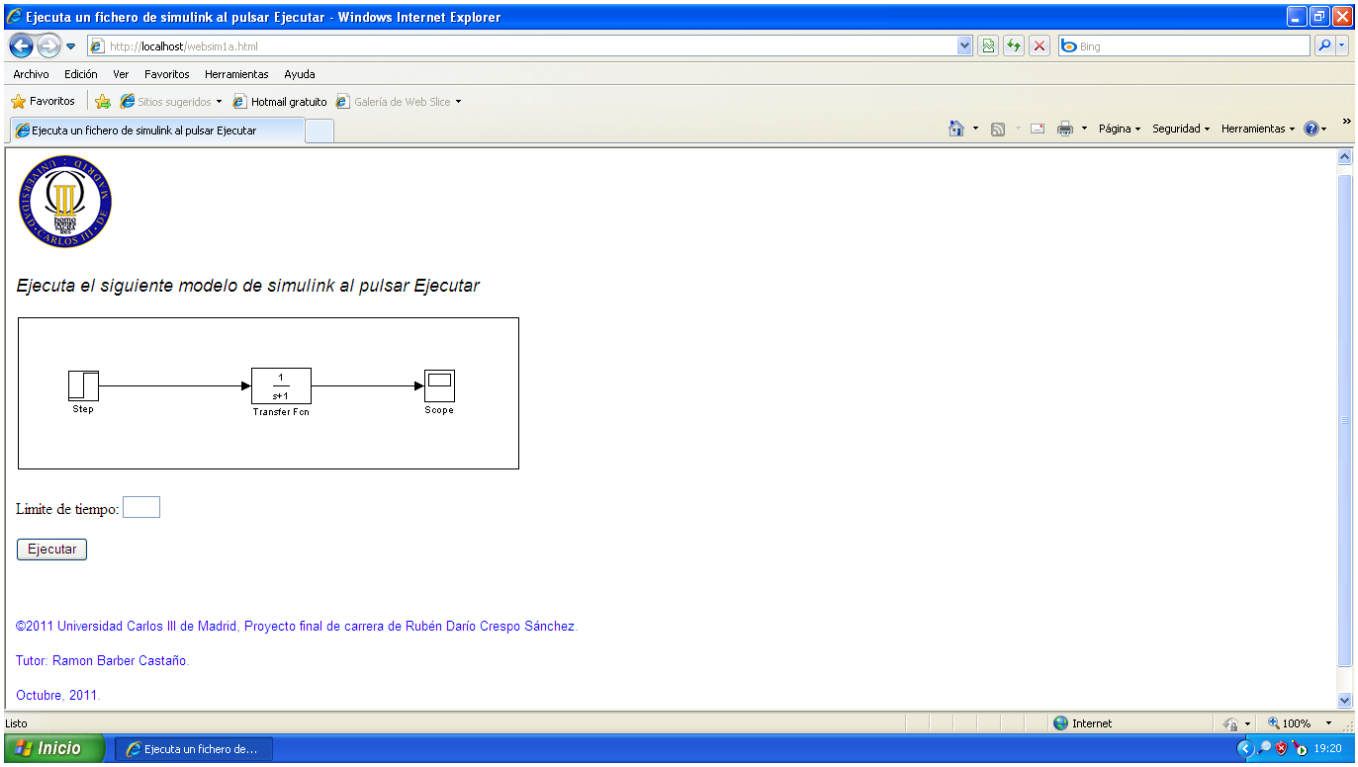


Figura 5.6. Página de entrada de datos del primer y segundo ejemplo

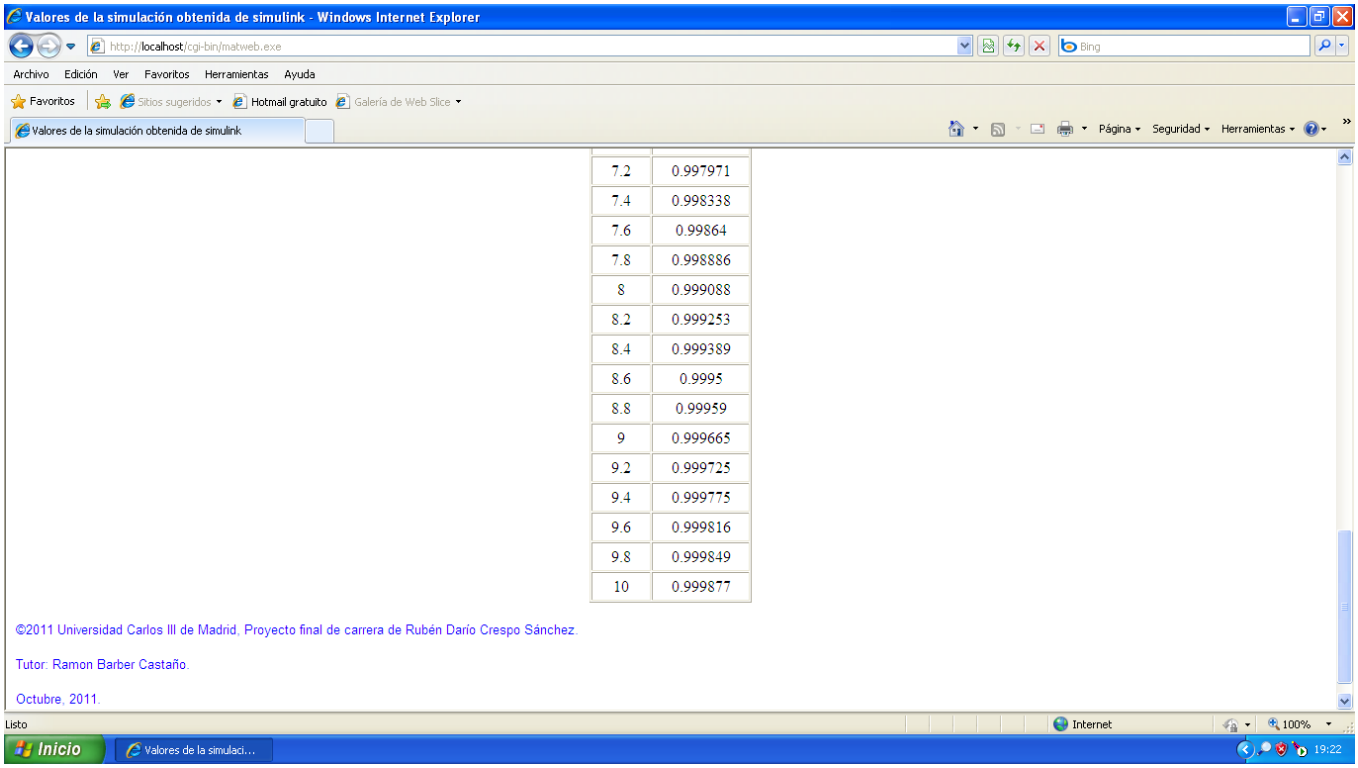


Figura 5.7. Página de resultados del primer ejemplo

5.2.2 Conexión mediante MATLAB Web Server con un modelo de Simulink sencillo con resultados gráficamente

Si en cambio se elige la segunda opción, aparecerá la misma página que en el primer ejemplo en la que se muestra el modelo a simular (Figura 5.6). En cambio, al pulsar el botón "Ejecutar" se ejecutará un modelo de Simulink que devolverá los resultados en una gráfica en vez de en una tabla como se puede ver en la Figura 5.8.

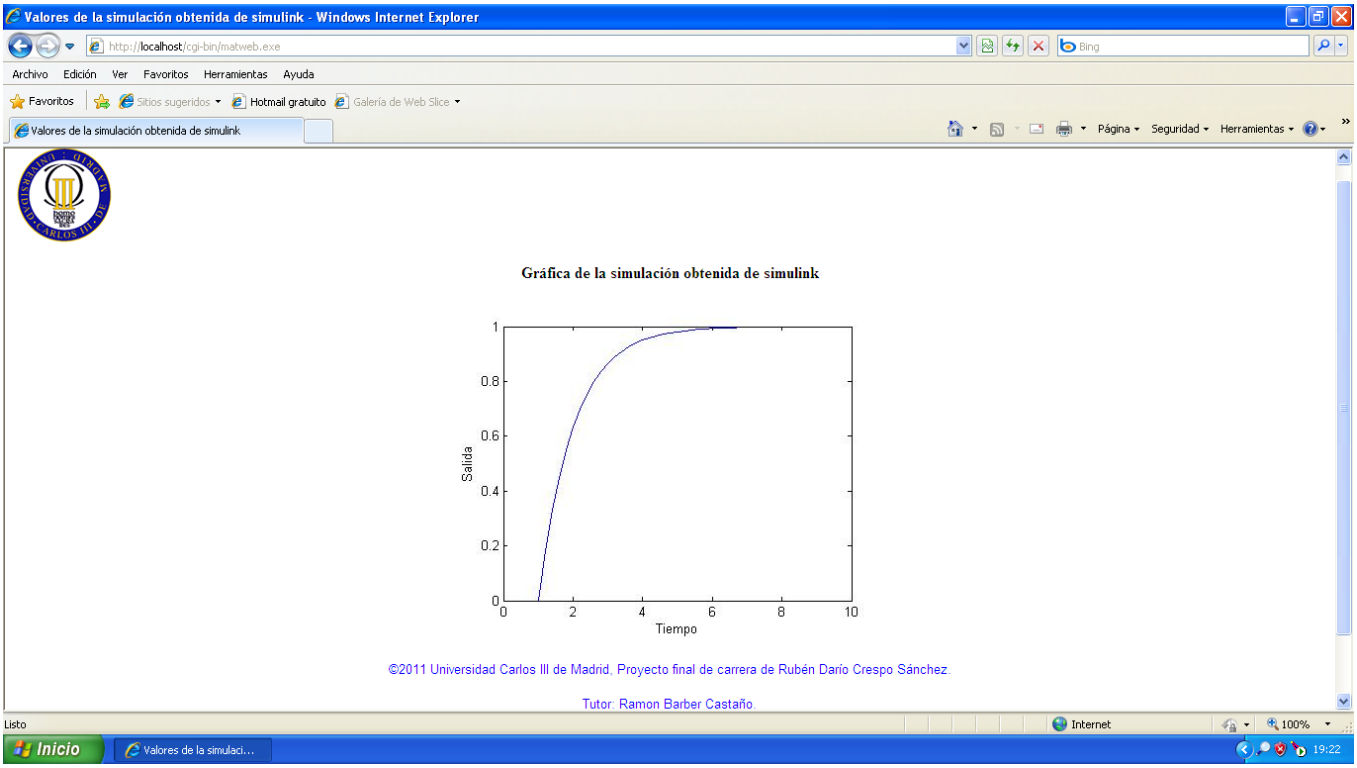


Figura 5.8. Página de resultados del segundo ejemplo

5.2.3 Conexión mediante MATLAB Web Server con un modelo de Simulink de mayor complejidad con resultados gráficamente

Si se elige la tercera opción, aparecerá una página en la que se muestra el modelo a simular, pero en este caso, al modelo a simular se le pueden introducir los parámetros del numerador y denominador de la función de transferencia, además del parámetro del tiempo. Esto se puede ver en la Figura 5.9. Al pulsar el botón "Ejecutar", se ejecutará el modelo de Simulink con los parámetros que se hayan indicado y se devolverán los resultados de manera gráfica como se muestra en la Figura 5.10.

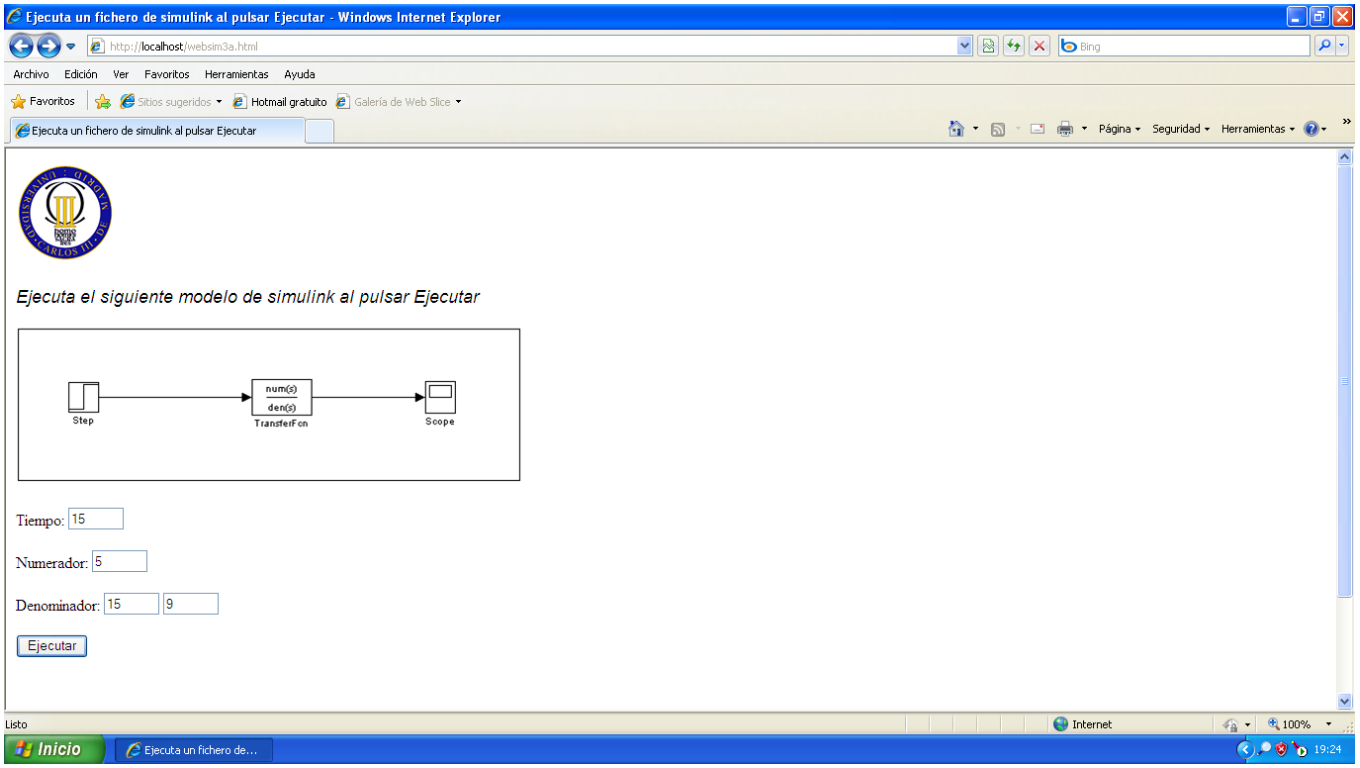


Figura 5.9. Página de entrada de datos del tercer ejemplo

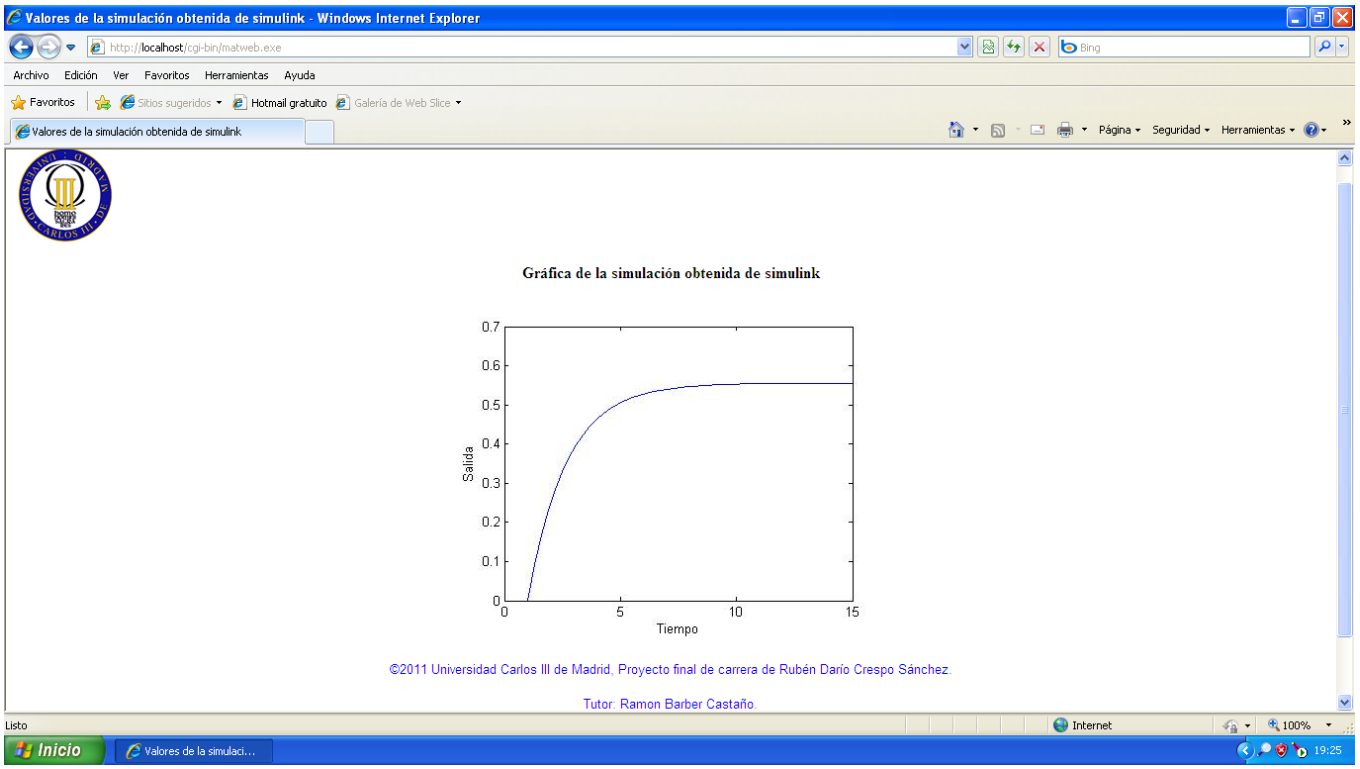


Figura 5.10. Página de resultados del tercer ejemplo

5.2.4 Conexión mediante MATLAB Web Server con el motor con resultados gráficamente

En la cuarta opción se consigue una conexión con el motor real del laboratorio. Se le puede indicar el tiempo de duración del ensayo, pero hay que tener en cuenta que el usuario tendrá que esperar ese tiempo a que termine el ensayo para poder visualizar los resultados, ya que es un ensayo en tiempo real, no una simulación como era el caso de las opciones anteriores.

Si se elige la cuarta opción, aparecerá una página en la que se muestra una foto del motor al que se conectará, y el espacio para indicar el tiempo del ensayo. Esto se puede ver en la Figura 5.11. Al pulsar el botón "Ejecutar", se ejecutará el motor durante el tiempo indicado. Una vez que haya finalizado el tiempo indicado con el motor funcionando, se devolverán los resultados de manera gráfica como se muestra en la Figura 5.12.

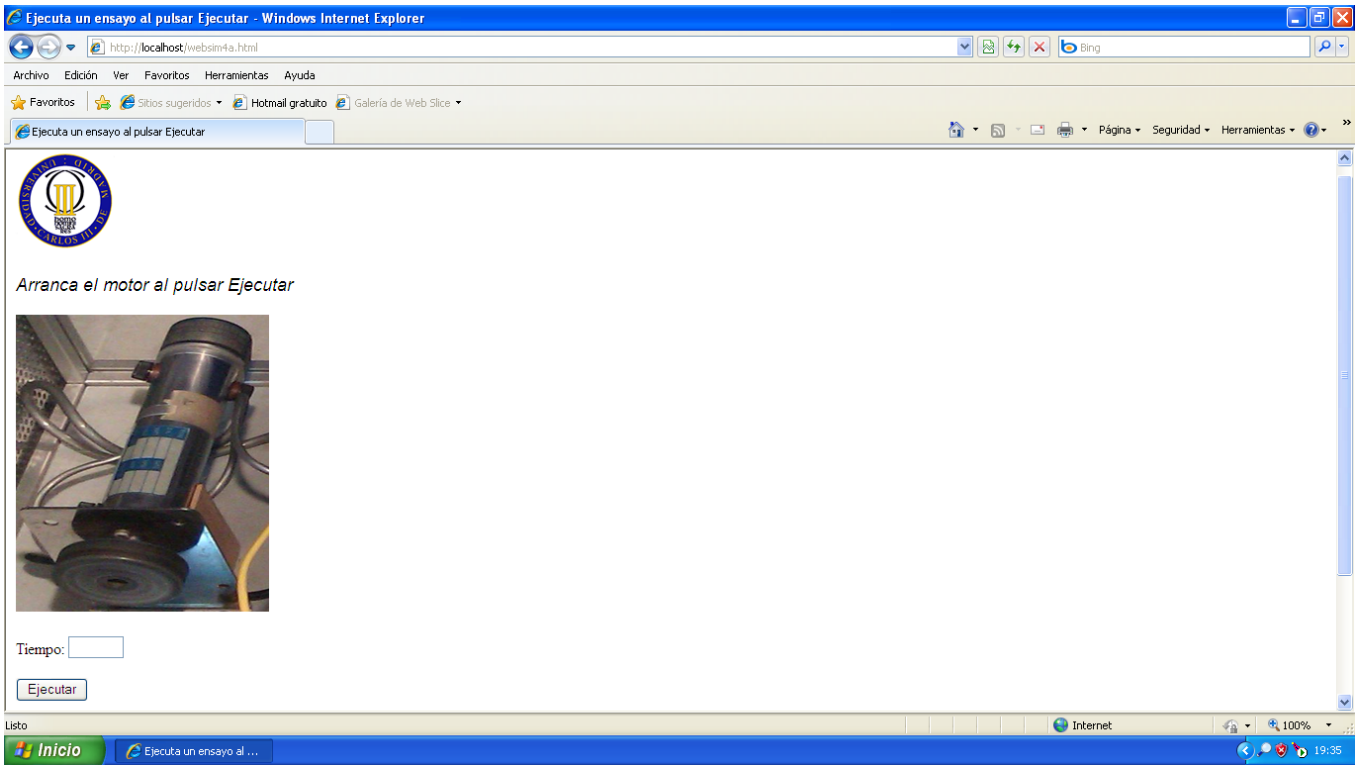


Figura 5.11. Página de entrada de datos del cuarto ejemplo

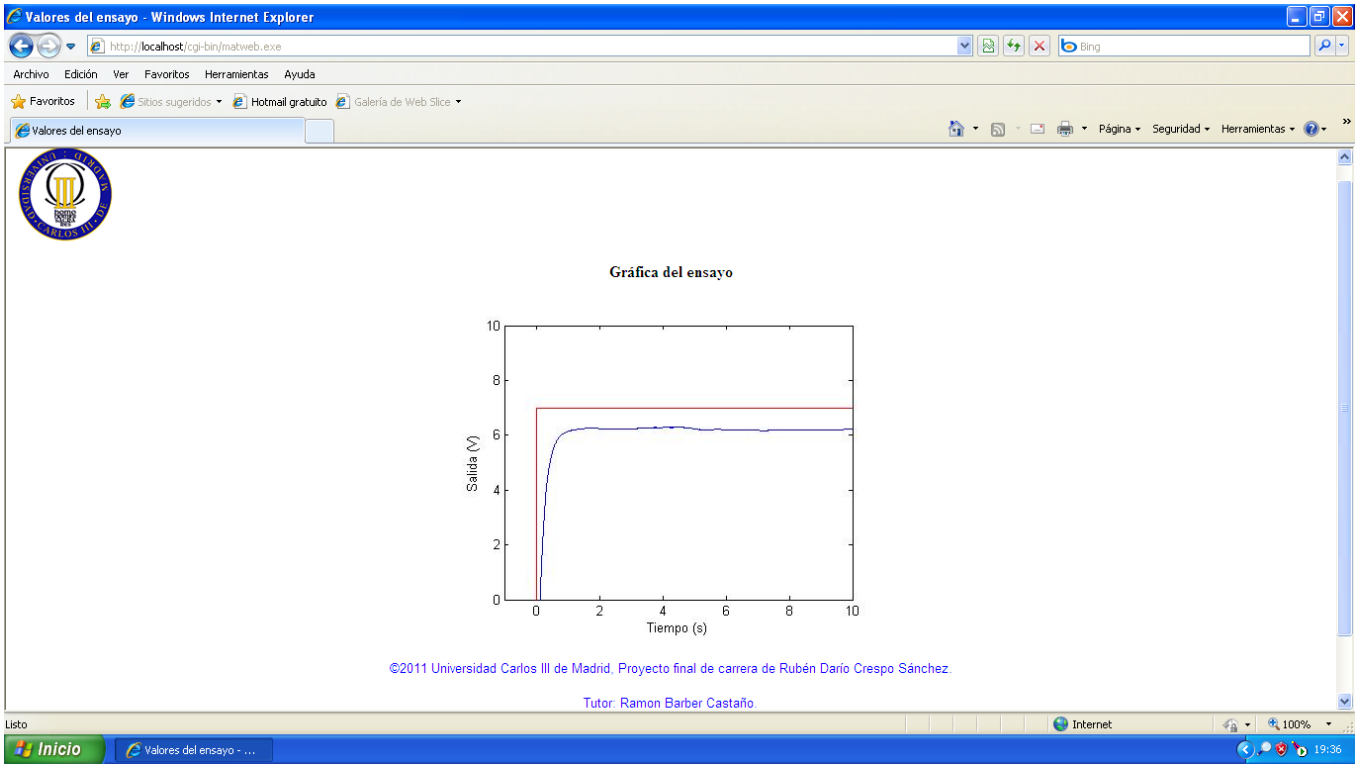


Figura 5.12. Página de resultados del cuarto ejemplo

5.2.5 Conexión mediante Easy Java Simulations con un modelo de Simulink sencillo con resultados gráficamente

Eligiendo la quinta opción, aparecerá una página web como la que se muestra en la Figura 5.13. Si se elige el primer ejemplo aparecerá una página web como la de la Figura 5.14 donde se puede observar la interfaz gráfica de Easy Java Simulations que se ha diseñado, en funcionamiento conectada con el modelo sencillo de Simulink.

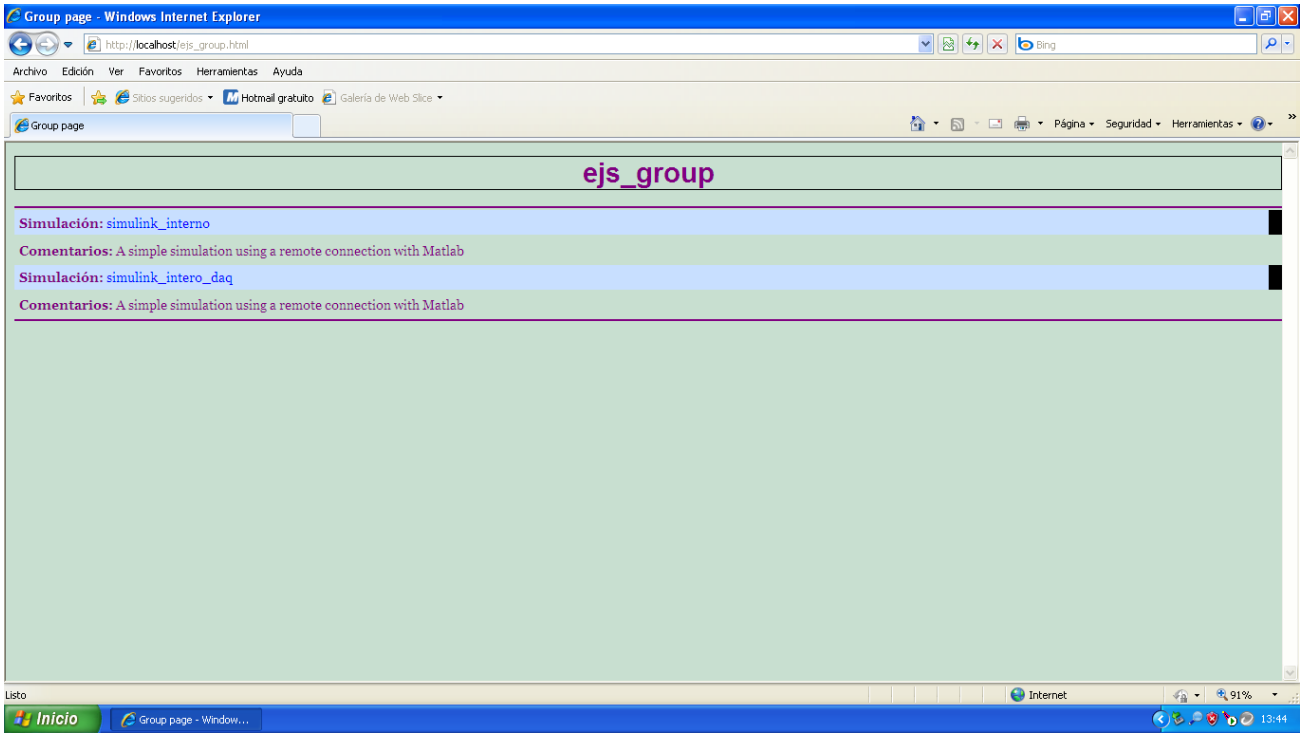


Figura 5.13. Página con las dos opciones de Easy Java Simulations

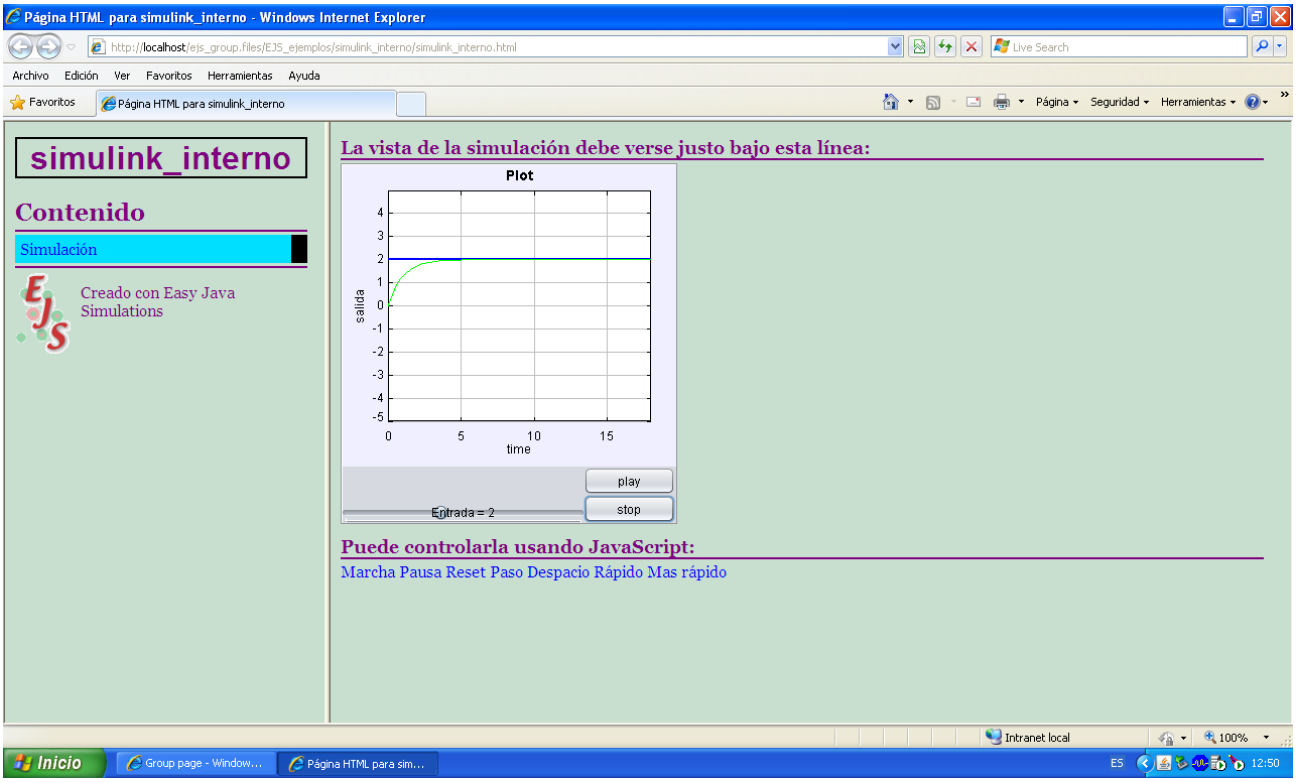


Figura 5.14. Página con los resultados del primer ejemplo de Easy Java Simulations

5.2.6 Conexión mediante Easy Java Simulations con el motor con resultados gráficamente

En cambio, si de entre las opciones de la quinta opción, se elige el segundo ejemplo aparecerá una página web como la de la Figura 5.15 donde se observa la interfaz gráfica de Easy Java Simulations que va mostrando los valores que va tomando la salida del motor.

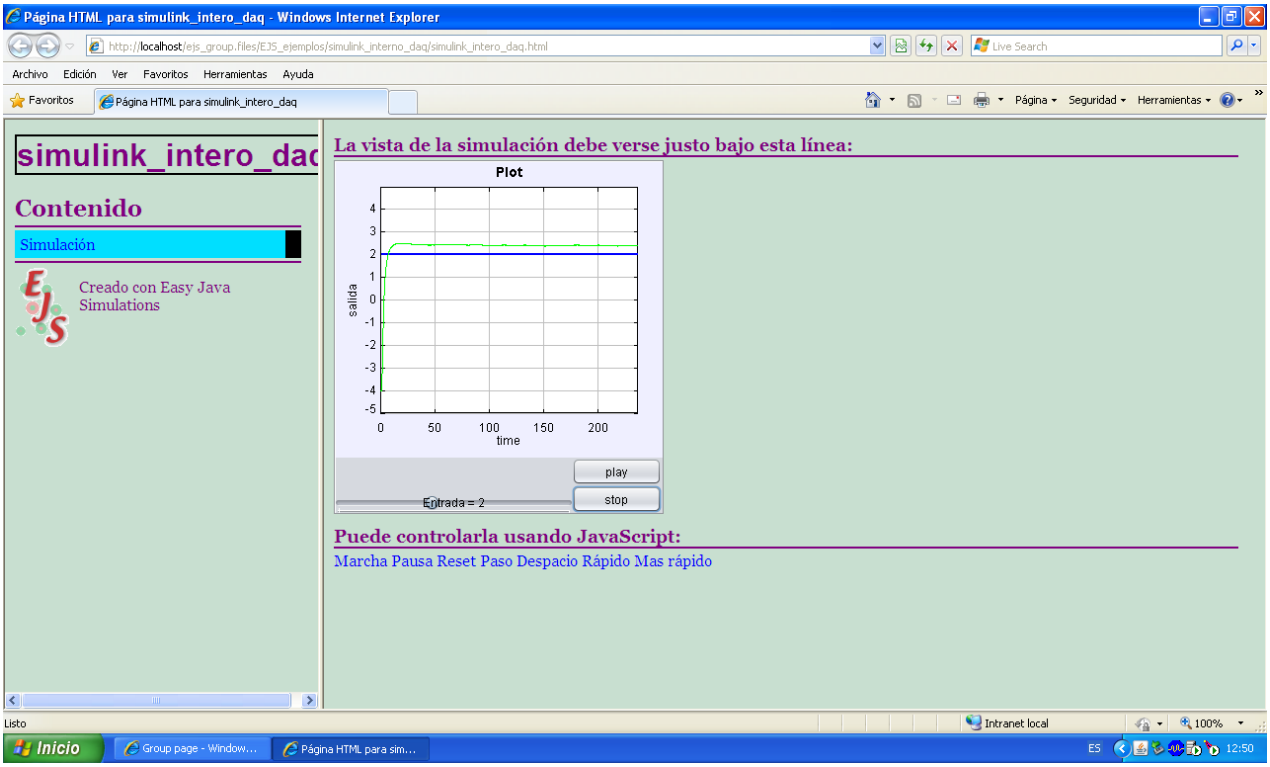


Figura 5.15. Página con los resultados del segundo ejemplo de Easy Java Simulations

Como se puede observar, la página web es muy básica y poco amigable. Pero este no es el objeto de este proyecto fin de carrera, hacer una página web es una labor relativamente sencilla (las páginas de Easy Java Simulations se generan automáticamente con el programa). Esto puede ser uno de los trabajos futuros de este proyecto, pero en este proyecto la finalidad y dificultad ha sido el conseguir conectarse de forma remota con MATLAB y Simulink, por eso no se ha dado importancia a la interfaz gráfica.

Otro aspecto es que los ejemplos se pueden complicar mucho más, para que se diera la posibilidad de introducir más parámetros. Esto tampoco ha sido objeto de este proyecto. En este proyecto se han desarrollado ejemplos básicos para explicar el

funcionamiento, pero se ha dejado preparado para que con futuros trabajos sea muy sencillo complicar los ejemplos. Es decir, se ha investigado como introducir parámetros a un bloque de Simulink, una vez que se tiene como introducir parámetros a un bloque resulta muy sencillo poder introducir parámetros a otros bloques diferentes. Resultó bastante complicado el poder pasar un vector del tipo [X X], pero se consiguió de manera que se pueden introducir los valores del vector uno por uno, y después desde MATLAB unirlos en un vector. En el caso de este proyecto es un vector 1x2, pero se puede realizar con un vector de otras dimensiones fácilmente.

5.3 Ejemplo de práctica remota

El fin de este proyecto es que los alumnos puedan realizar prácticas remotas ya sea realizando simulaciones como ensayos reales. Desde el punto de visto pedagógico tiene más importancia el poder realizar en ensayo real, para que el alumno de ingeniería pueda identificar una planta real.

Por este motivo, en este apartado se ha realizado lo que podría ser la primera práctica remota de alguna asignatura de control de diferentes titulaciones. Se ha optado por realizar la práctica usando MATLAB Web Server, pero se podría realizar del mismo modo con Easy Java Simulations.

La práctica a realizar es: Identificación de un motor de corriente continua remotamente.

Para realizar la identificación de un motor primero hay que tener claro el sistema con el que se va a trabajar. En la Figura 5.16 se muestra el sistema real del laboratorio. Se puede observar el motor en la parte izquierda, conectado a la placa de adquisición de datos que está en la parte derecha. Las conexiones se realizan por medio de los conectores que hay en el centro de la maqueta. Desde el PC se manda un dato digital que se convierte en una señal analógica que se conecta a la entrada del motor a través de la tarjeta de adquisición de datos. La salida analógica que mide la velocidad del motor se conecta con la entrada analógica de la tarjeta de adquisición de datos, que se convertirá en un dato digital que procesará el PC. Es necesario unir todas las masas para fijar la referencia. Se unen las masas de los dos canales que se usan de la tarjeta de adquisición de datos, y del lazo de realimentación ya que la respuesta del motor se va a analizar en cadena abierta.

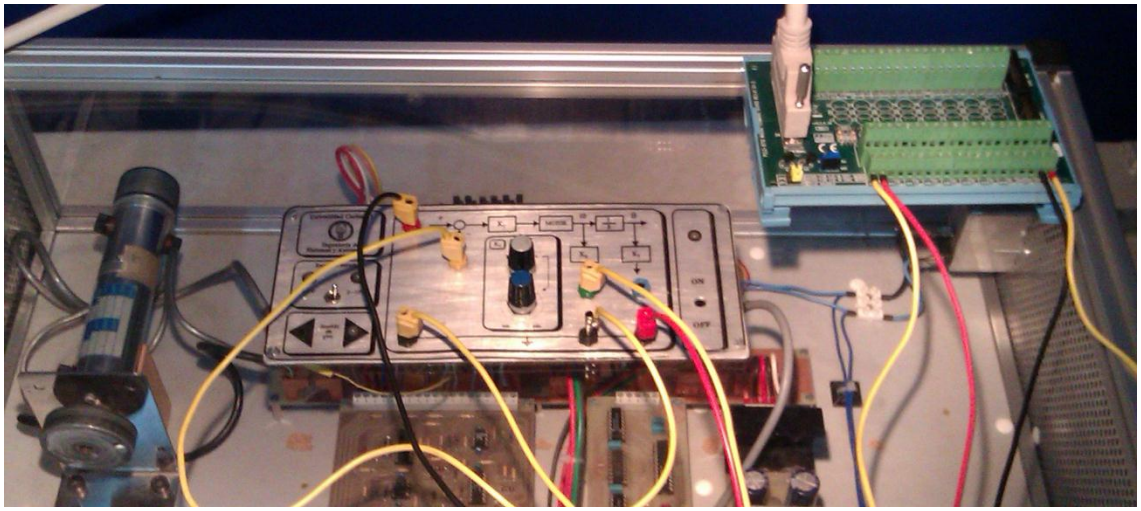


Figura 5.16. Sistema real del laboratorio

Con el fin de entender mejor el funcionamiento, es conveniente tener un esquema conceptual desarrollado a partir del sistema real. En la Figura 5.17 se muestra el sistema de control simplificado del sistema real del laboratorio.

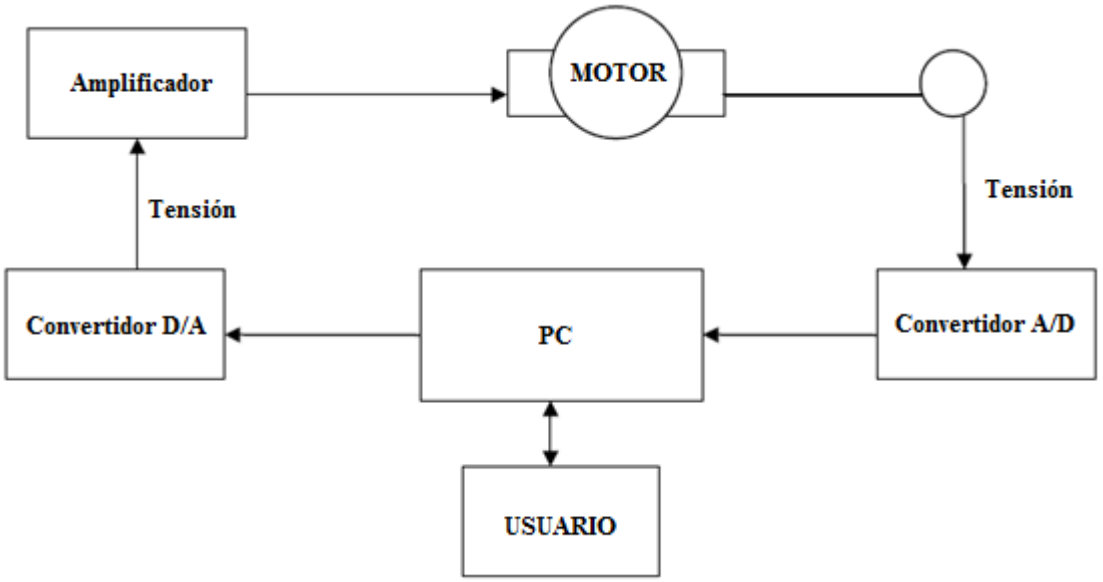


Figura 5.17. Sistema de control digital de un motor de corriente continua

Una vez se tiene claro el sistema sobre el que se va a actuar, hay que extraer la función de transferencia del sistema físico. El sistema motor está esquematizado en la Figura 5.18. De ahí se extrae la función de transferencia en lazo abierto del motor con salida en velocidad. La Ecuación 5.1 determina la función de transferencia de este sistema.

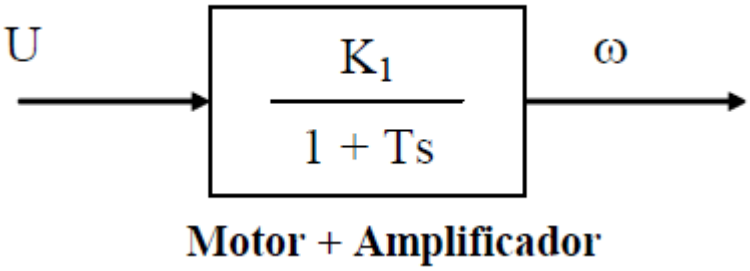


Figura 5.18. Sistema motor con salida en velocidad

$$G(s) = \frac{\omega(s)}{U(s)} = \frac{K_1}{(1 + Ts)}$$

Ecuación 5.1. Función de transferencia del sistema

Dónde:

K_1: ganancia del sistema

T: constante de tiempo del motor

Para realizar la identificación de los parámetros del sistema es necesario realizar el análisis de la respuesta temporal del sistema en lazo abierto. Para ello, habría que entrar en la página web http://docenciaisa.uc3m.es/, y dentro de la web navegar hasta encontrar el ejemplo de "Ejecución del motor". Este ejemplo lanza un escalón al motor durante el tiempo que se especifique. Una vez completado el ensayo con el motor real, se devolverá una gráfica en la página web como la de la Figura 5.19, en la que se pueden medir ciertos parámetros para identificar el motor.

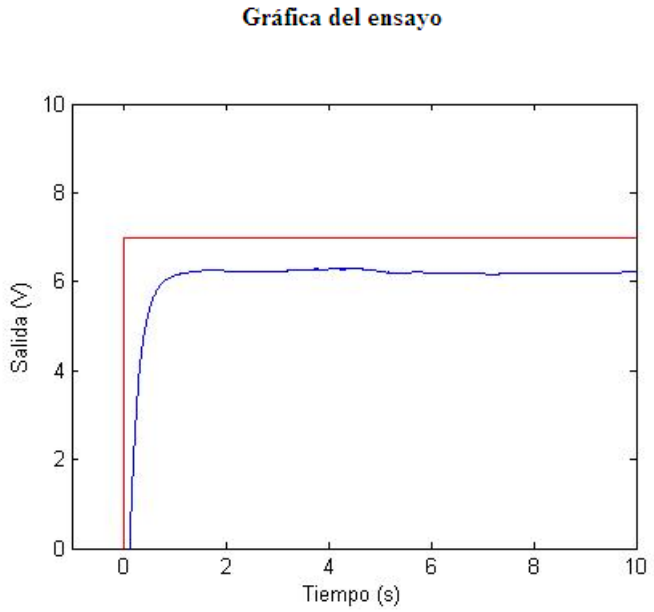


Figura 5.19. Gráfica del ensayo realizado

Es necesario medir le tensión en régimen permanente (V_o) para obtener la ganancia (K_1). Sabiendo que la tensión de entrada es V_i = 7V y la tensión de salida medida de la gráfica es V_o = 6,2V, la ganancia (K_1) se obtiene aplicando la Ecuación 5.2.

$$K_1 = \frac{V_o}{V_i} = \frac{6,2}{7} = 0,89$$

Ecuación 5.2. Fórmula para obtener la ganancia

A partir de la gráfica también se obtiene la constante de tiempo, calculando el 63,21% del valor de la salida en régimen permanente y viendo dónde corta a la gráfica. El 63,21% de V_o es 3,9V, que corresponde con 0,38s, como se puede observar en la Figura 5.20.

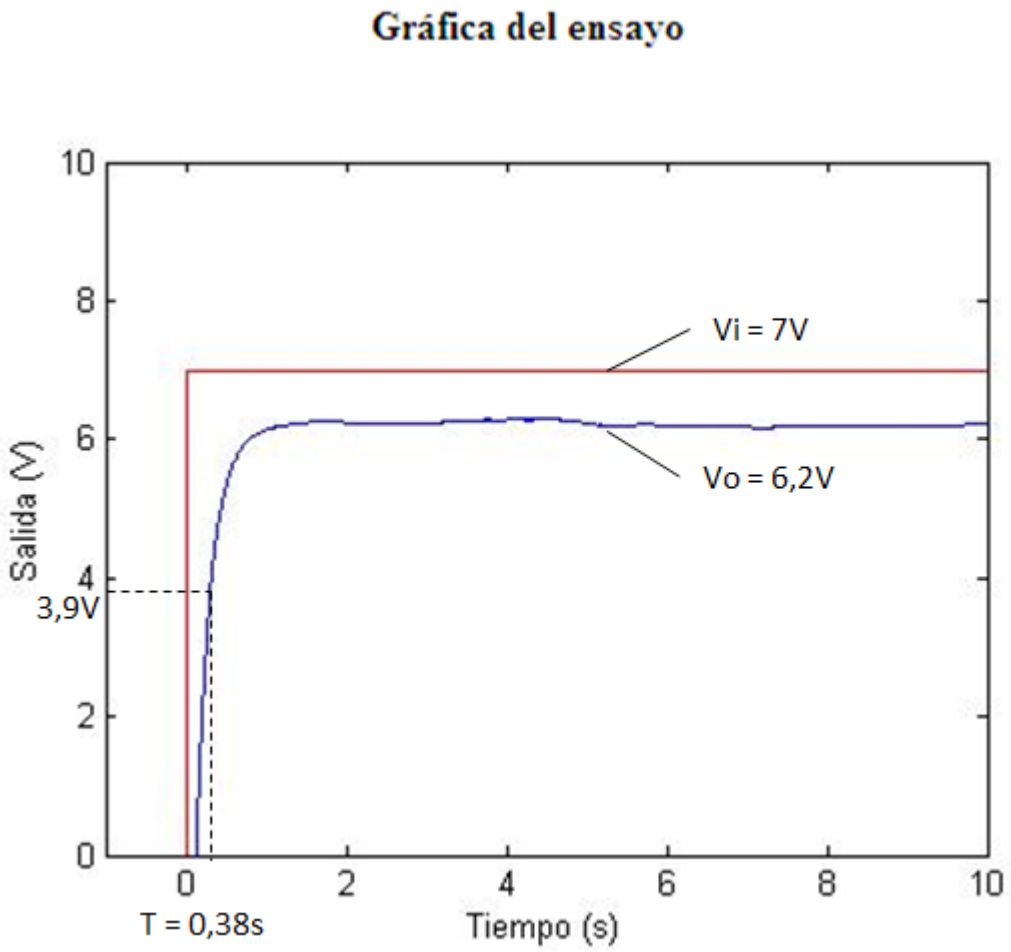


Figura 5.20. Gráfica del ensayo realizado con los resultados

Con esto ya se tiene identificado el sistema motor. Su función de transferencia queda como se puede ver en la Ecuación 5.3.

$$G(s) = \frac{\omega(s)}{U(s)} = \frac{0{,}89}{(1 + 0{,}38s)}$$

Ecuación 5.3. Función de transferencia del sistema con los valores medidos remotamente

Después de realizar la práctica de identificación del motor, se podría complicar el ejemplo primero con otro tipo de sistemas. Estos ejemplos podrían ser las siguientes prácticas.

Capítulo 6

Conclusiones

6 Conclusiones

En este apartado se van a extraer las conclusiones a las que se ha llegado con este proyecto después de haberlo desarrollado y probado. Además, también se van a describir los posibles trabajos futuros que se pueden realizar teniendo como base este proyecto. Estos trabajos futuros están derivados directamente de las conclusiones.

6.1 Conclusión

La principal conclusión que se extrae de este proyecto es la viabilidad y utilidad de un laboratorio remoto. Además, lo conseguido en este proyecto se puede ampliar de muchas maneras desarrollando todo tipo de prácticas remotas con todo tipo de plantas reales. Este proyecto ha sido la base para conseguir la funcionalidad básica, es decir, conseguir la conectividad a una planta real remotamente.

A continuación se exponen las conclusiones de las tecnologías usadas:

MATLAB Web Server es una tecnología que ya no incorpora MATLAB en sus versiones más recientes. No obstante, es una tecnología muy útil que se puede seguir usando. MATLAB Web Server permite realizar varias simulaciones a la vez y devuelve los datos con gran rapidez. Además, usando la toolbox Data Acquisition permite la conectividad con la planta real. Así se puede conectar con cualquier planta sea del tipo que sea.

Easy Java Simulations es una herramienta relativamente moderna con muchas posibilidades. Es muy fácil de usar y permite la conexión de forma remota por lo que se pueden crear muchos ejemplos de prácticas de una manera muy sencilla.

Con ninguna de las dos tecnologías es posible conectarse remotamente con un modelo de Simulink en modo externo.

La conexión con MATLAB y Simulink son imprescindibles en las soluciones que se han buscado ya que este proyecto está orientado a la realización de prácticas remotas por parte de alumnos de ingenierías. Las escuelas de ingeniería utilizan estos programas.

6.2 Trabajos futuros

Existen numerosos trabajos futuros a poder realizar sobre este proyecto, ya que en este proyecto se ha conseguido la conectividad a una planta real remotamente. En este

proyecto se ha conseguido la base de funcionamiento. A partir de la base, se puede trabajar por diferentes vías. Las mejoras que se pueden hacer se detallan a continuación:

- Conseguir la conexión con el motor mediante comandos de la Data Acquisition toolbox usando Easy Java Simulations, al igual que se consiguió en este proyecto con MATLAB Web Server.
- Complicar los ejemplos ya creados. Bien sea incluyendo más bloques en los modelos de Simulink para poder visualizar otros fenómenos, poder introducir el número de términos que se quiera en la función de transferencia, crear otros ejemplos completamente diferentes, etc. El límite lo impone Simulink. La idea es poder crear nuevos ejemplos y prácticas lo más didácticos posibles para que los alumnos puedan aprender más.
- Permitir la posibilidad de que MATLAB guarde los resultados de las simulaciones o ensayos en un fichero que el alumno pueda descargar a su PC. Con los datos en un archivo el alumno puede exportarlos, por ejemplo a Excel, para realizar un tratamiento de los datos. Así el alumno puede analizar los datos de diferentes formas.
- Otro trabajo futuro muy importante es diseñar una página web que sea cómoda para el usuario, en la que poder indicar más parámetros de manera más sencilla. Una web que muestre la gráfica de resultados en la misma página donde se introducen los datos, por lo tanto se puede modificar un parámetro y ver cómo cambia de los resultados anteriores a los nuevos. Una web que muestre los resultados según se va realizando el ensayo incorporando la imagen del motor en funcionamiento (o de la planta del laboratorio) en todo momento mediante una webcam.
- Con el fin de poder medir mejor los parámetros de las gráficas, se podría crear una gráfica cuadriculada, que se pudiese seleccionar y ampliar la zona de mayor interés.
- Crear ejemplos de prácticas que no usen un motor sino cualquier otro tipo de planta real, como un secador, un tanque, etc.

El sistema creado en este proyecto es un sistema que puede y debe estar en continuo desarrollo en función de cómo evolucionen el tipo de prácticas que realicen los alumnos y en función de las necesidades de los alumnos que son para quienes da funcionalidad esta idea.

Como conclusión final, los trabajos realizados en este proyecto no sólo aportan soluciones que muestran la viabilidad de utilizar laboratorios remotos, sino que abre muchas posibilidades de continuar con estudios, investigaciones, desarrollos y pruebas destinados a la validación de los sistemas y a la aceptación por parte de profesores y alumnos.

Bibliografía

Bibliografía

MathWorks. (Julio de 2010). Obtenido de http://www.mathworks.es

CAE Software / MATLAB. (Julio de 2011). Obtenido de http://www.kxcad.net/cae_MATLAB/toolbox/rtwin/ug/f7057.html

DeVry University Keller. (Enero de 2011). Obtenido de http://www.tp.devry.edu/PDFs/Using_Matlab_Simulink.pdf

Easy Java Simulations. (Julio de 2011). Obtenido de http://fem.um.es/Ejs/

Enlaces interesantes: laboratorios virtuales y/o remotos. (Septiembre de 2011). Obtenido de http://www.cea-ifac.es/wwwgrupos/educontrol/enl-labs.html

Laboratorio virtual de servomecanismos. (Julio de 2011). Obtenido de http://www.esi2.us.es/~limon/LabVirServo.html

MATLAB Newsreader. (Agosto de 2011). Obtenido de http://www.mathworks.de/matlabcentral/newsreader/

Monografias.com (Real Time Windows Target). (Septiembre de 2011). Obtenido de http://www.monografias.com/trabajos22/aplicaciones-matlab/aplicaciones-matlab.shtml

Using Ejs to run Simulink models in an interactive way for version 3.4. (Agosto de 2011). Obtenido de http://www.um.es/fem/Download/Ejs/EjsAndSimulink.pdf

Wiki pages for Easy Java Simulations. (Julio de 2011). Obtenido de http://www.um.es/fem/EjsWiki

A. Valera, M. Vallés, J.L. Díez. (2005). Simulación y control de procesos físicos de forma remota.

Antonio Giménez, Ramón Barber. (2010). Práctica 1 de Ingeniería de Control.

Cilento, A. (2007). *Sistemas de control de motores en Tiempo Real mediante Matlab.*

David Jofre Hernández, Ramon Costa Castelló, Luis Basañez Villaluenga. (s.f.). Laboratorio remoto de Automática: Plantas de variable continua.

Farias Castro, G. (Octubre de 2011). Reunión en el laboratorio de la Universidad Carlos III de Madrid para aportarnos sus conocimientos sobre Easy Java Simulations. (R. I. Barber Castaño, & R. D. Crespo Sánchez, Entrevistadores)

Fernández, S. M. (2007). *WLAB – Sistema de control remoto en tiempo real de equipos de laboratorio.*

G. Farias, F. Esquembre, J. Sánchez, S. Dormido. (2006). Laboratorios virtuales remotos usando Easy Java Simulations y Simulink.

Gonzalo Farias, Robin De Keyser, Sebastián Dormido, Francisco Esquembre. (2010). Developing Networked Control Labs: A Matlab and Easy Java Simulations Approach.

Inc., T. M. (1999). *Matlab Web Server.*

Introducción a Simulink. (s.f.).

Javier Aracil, Fabio Gómez-Estern. (2007). *Introducción a Matlab y Simulink.*

Javier García de Jalón, José Ignacio Rodríguez, Jesús Vidal. (2005). *Aprenda Matlab 7.0.*

Luis M. Jiménez, Rafael Puerto, Óscar Reinoso, César Fernández, Ramón Ñeco. (2005). Recolab: Laboratorio remoto de control utilizando Matlab y Simulink.

Madrid, U. P. (2007). *Simulación de sistemas con Simulink.*

Martínez Bejarano, R. (2006). *Autolab : aplicación para el autoaprendizaje a través de Internet : laboratorio de sistemas basados en microprocesador.*

Oviedo, U. d. (2006). *Simulación de sistemas de control continuos con Matlab y Simulink.*

Palacios, R. (Julio de 2011). Consulta sobre el uso de Matlab Web Server. (R. D. Crespo Sánchez, Entrevistador)

Pizarro Jiménez, P. (2008). *Laboratorio de autoaprendizaje remoto para sistemas basados en ARM7 : VirtuaLab ARM7.*

Puerto, R. (Septiembre de 2011). Consulta sobre el uso de Matlab Web Server. (R. I. Barber Castaño, Entrevistador)

Puerto, R. (Julio de 2011). Consulta sobre el uso de Matlab Web Server. (R. D. Crespo Sánchez, Entrevistador)

R. Pastor, J. Sánchez, S. Dormido. (2005). Web-Based virtual lab and remote experimentation using Easy Java Simulations.

Rafael Palacios, Ramón Rodríguez Pecharromán. (2005). Acceso remoto a ensayos de control en tiempo real basados en Matlab.

Rafael Puerto, Luis M. Jiménez, Óscar Reinoso, César Fernández, Ramón Ñeco. (2004). *Remote control laboratory using Matlab and Simulink: Application to a DC motor model.*

Rafael Puerto, Óscar Reinoso, Ramón Ñeco, Nicolás García, Luis M. Jiménez. (2001). Remote lab for control applications usign Matlab.

Rosa, A. R. (2006). *WLAB – Sistema de control remoto en tiempo real de equipos de laboratorio.*

Presupuesto

Presupuesto

UNIVERSIDAD CARLOS III DE MADRID
Escuela Politécnica Superior

PRESUPUESTO DE PROYECTO

1.- Autor: Rubén Darío Crespo Sánchez

2.- Departamento: Ingeniería de Sistemas y Automática

3.- Descripción del Proyecto:

- Titulo
- Duración (meses) **6**

Tasa de costes Indirectos: **20%**

4.- Presupuesto total del Proyecto (Euros):

Veintiséis mil cuarenta y un euros

5.- Desglose presupuestario (costes directos)

PERSONAL

Apellidos y nombre	Categoría	Dedicación (hombres mes) [a)]	Coste hombre mes	Coste (Euros)
Ramón Ignacio Barber Castaño	Ingeniero Sénior	2	4.289,54	8.579,08
Rubén Darío Crespo Sánchez	Ingeniero	4	2.694,39	10.777,56
Rubén Darío Crespo Sánchez	Administrativo	1	1.774,32	1.774,32
	Hombres mes	**7**	**Total**	21.130,96

[a)] 1 Hombre mes = 131,25 horas. Máximo anual de dedicación de 12 hombres mes (1575 horas)
Máximo anual para PDI de la Universidad Carlos III de Madrid de 8,8 hombres mes (1.155 horas)

EQUIPOS

Descripción	Coste (Euros)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable (Euros) [d)]
PC	399,00	100	6	60	39,90
Tarjeta de adquisición de datos	284,83	100	5	60	23,74
Maqueta con motor y circuitería	291,87	100	5	60	24,32
				Total	87,96

[d)] Fórmula de cálculo de la Amortización:

$$\frac{A}{B} x C x D$$

A = nº de meses desde la fecha de facturación en que el equipo es utilizado
B = periodo de depreciación (60 meses)
C = coste del equipo (sin IVA)
D = % del uso que se dedica al proyecto (habitualmente 100%)

SUBCONTRATACIÓN DE TAREAS

Descripción	Empresa	Coste imputable (Euros)
Solicitación de ayuda a profesor	Universidad Miguel Hernández	20
Solicitación de ayuda a profesor	Universidad Pontificia de Comillas	30
Solicitación de ayuda a profesor	Universidad Nacional de Educación a Distancia	120,00
	Total	170,00

OTROS COSTES DIRECTOS DEL PROYECTO[e)]

Descripción	Empresa	Costes imputable (Euros)
Viajes	Particular	312,00
	Total	312,00

[e)] Este capítulo de gastos incluye todos los gastos no contemplados en los conceptos anteriores, por ejemplo: fungible, viajes y dietas, otro, etc.

6.- Resumen de costes

Presupuesto Costes Totales	Costes Totales (Euros)
Personal	21.131
Amortización	88
Subcontratación de tareas	170
Costes de funcionamiento	312
Costes Indirectos	4.340
Total	**26.041**

El presupuesto total de este proyecto asciende a:

VEINTISÉIS MIL CUARENTA Y UN EUROS

Madrid, a 18 de Octubre de 2011

Fdo.: Rubén Darío Crespo Sánchez

Ingeniero Industrial

Anexos

Anexos

Hojas de características de la tarjeta de adquisición de datos

PCI-1711
PCI-1711L

100 kS/s, 12-bit, 16-ch S.E. Inputs Low-cost Multifunction Card

100 kS/s, 12-bit, 16-ch S.E. Inputs Low-cost Multifunction Card w/o AO

Features

- 16 single-ended analog inputs
- 12-bit A/D converter, with up to 100 kHz sampling rate
- Programmable gain for each input channel
- Automatic channel/gain scanning
- On-board 1K samples FIFO buffer
- Two 12-bit analog output channels (Only for PCI-1711)
- 16 digital inputs and 16 digital outputs
- Programmable pacer/counter

Introduction

PCI-1711 and PCI-1711L are powerful, but low-cost multifunction cards for the PCI bus. PCI-1711 comes with 2 analog output channels, while the PCI-1711L doesn't. Thus, PCI-1711L represents a cost saver for those that do not need analog output.

Specifications

Analog Input

- **Channels** 16 Single-Ended
- **Resolution** 12-bit
- **FIFO Size** 1K samples
- **Sampling Rate*** 100 kS/s max.

Input range and Gain List		Gain	1	2	4	8	16
		Input	±10 V	±5 V	±2.5 V	±1.25 V	±0.625 V
Drift (ppm/°C)		Gain	1	2	4	8	16
		Zero	15	15	15	15	15
		Gain	25	25	25	30	40
Small Signal Bandwidth for PGA		Gain	1	2	4	8	16
		Bandwidth	4.0 MHz	2.0 MHz	1.5 MHz	0.65 MHz	0.35 MHz

- **Max. Input Overvoltage** 20 V
- **Input Protect** 30 Vp-p
- **Input Impedance** 2 MΩ/5 pF
- **Trigger Mode** Software, On-board Programmable Pacer or external

Accuracy	DC	INLE: ±0.5 LSB
		Monotonicity: 12 bits
		Offset error: Adjustable to zero
		Gain error: 0.005% FSR (Gain=1)
	AC	SNR: 68 dB
		ENOB: 11 bits

Programmable Counter / Timer

- **Channels** 1
- **Resolution** 16-bit
- **Compatibility** TTL level
- **Base Clock** 10 MHz
- **Max. Input Frequency** 10 MHz

Note:
The sampling rate and throughput depends on the computer hardware architecture and software environment. The rates may vary due to programming language, code efficiency, CPU utilization and so on.

Analog Output (only for PCI-1711)

- **Channels** 2
- **Resolution** 12-bit

Output Range (Internal & External Reference)	Internal Reference	0 ~ +5 V, 0 ~ +10 V
	External Reference	0 ~ +x V @ -x V (-10 ≤ x ≤ 10)
Accuracy	Relative	±1/2 LSB
	Differential Non-linearity	±1/2 LSB

- **Gain Error** Adjustable to zero
- **Slew Rate** 11 V/μs
- **Drift** 40 ppm/° C
- **Driving Capability** 3 mA
- **Throughput** PC dependent, Software update (direct AO)
- **Output Impedance** 0.81 Ω
- **Settling Time** 26 μs (to ±1/2 LSB of FSR)
- **Reference Voltage** Internal -5 or -10 V
  External -10 or +10 V

Digital Input / Output

Input Channels		16
Input Voltage	Low	0.8 V max.
	High	2.0 V max.
Output Channels		16
Output Voltage	Low	0.8 V max.@ 8.0 mA (sink)
	High	2.0 V min.@ -0.4 mA (source)

General

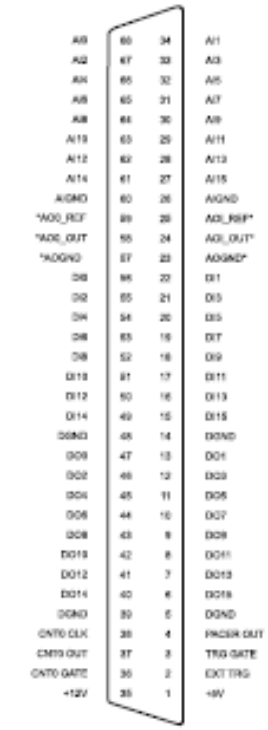
I/O Connector Type		68-pin SCSI-II female	
Dimensions		175 x 100 mm (6.9" x 3.9")	
Power Consumption	Typical	PCI-1711	PCI-1711L
		+5 V @ 850 mA	+5 V @ 700 mA
	Max.	+5 V @ 1.0 A	
Temperature	Operation	0 ~ 60° C (32 ~ 140° F) (refer to IEC 68-2-1, 2)	
	Storage	-20 ~ 70° C (-4 ~ 158° F)	
Relative Humidity		5 % ~ 95 % RH non-condensing (refer to IEC 68-2-3)	

ADVANTECH Plug-in DA&C Cards

Ordering Information

- **PCI-1711** 100 kS/s, 12-bit, 16-ch S.E. Inputs Low-cost Multifunction Card, user's manual and driver CD-ROM. (cable not included)
- **PCI-1711L** 100 kS/s, 12-bit, 16-ch S.E. Inputs Low-cost Multifunction Card w/o analog output, user's manual and driver CD-ROM. (cable not included)
- **PCLD-8710** Industrial Wiring Terminal Board with CJC circuit for DIN-rail mounting (cable not included)
- **PCL-10168** 68-pin SCSI-II cable with male connectors on both ends and special shielding for noise reduction, 1 and 2 m
- **ADAM-3968** 68-pin SCSI-II Wiring Terminal Board for DIN-rail Mounting

Pin Assignments



*: Pins 23~25 and pins 57~59 are not defined for PCI-1711L

Feature Details

Plug & Play Function

PCI-1711 and PCI-1711L fully comply complies with the PCI Specification Rev 2.1. and thus are Plug & Play devices. During card installation, it is virtually unnecessary to set any jumpers or DIP switches. Instead, all bus-related configurations such as base I/O address and interrupts are conveniently taken care of by the Plug & Play function.

Flexible Input Types and Range Settings

PCI-1711 and PCI-1711L feature an automatic channel/gain scanning circuit. This circuit design controls multiplexer switching during sampling. You can set different gain values for each channel according to your needs for the corresponding range of input voltages. The gain values thus selected are stored in the SRAM. This flexible design enables multi-channel and high-speed sampling for high-performance data acquisition (up to 100 kS/s).

On-board FIFO Memory

PCI-1711 and PCI-1711L provide an onboard FIFO (First In First Out) memory buffer, storing up to 1 K A/D samplings. You can either enable or disable the interrupt request feature of the FIFO buffer. While the interrupt request for FIFO is enabled, you can further specify whether the interrupt request will be sent whenever one sampling takes place or when the FIFO buffer is half saturated. This feature enables a continuous high-speed data transfer with more predictable performance on Windows systems.

Onboard Programmable Counter

PCI-1711 and PCI-1711L are equipped with a programmable counter, which can serve as a pacer trigger for A/D conversions. The counter chip is an 82C54 or equivalent, which incorporates three 16-bit counters on a 10 MHz clock. One of the three counters is used as an event counter for input channels. The other two are cascaded into a 32-bit timer for pacer triggering.

Applications

- Process monitoring and control
- Transducer and sensor measurement
- Multi-channel DC voltage measurement

Código de websim2.m

En este apartado se incluye el código del archivo websim2.m como ejemplo de código de los demás ficheros. La estructura es igual en los demás archivos de websim#.m, solamente se cambian las líneas de código que calculan los resultados. Estas líneas de código están detalladas en el documento. Se ha escogido este archivo porque devuelve los datos gráficamente que es el modo en que se han programado todos los archivos excepto websim1.m que los devuelve mediante una tabla.

```
function retstr = websim2(instruct, outfile)

%
%   INSTRUCT is a structure created by the matweb program.
%   It contains fields corresponding to the HTML form fields
%   in the HTML form, websim1.html.  In websim1.html there
%   is a hidden field, called INSTRUCT.MLMFILE that references
%   this websim.m M-file.
%
%   ©2011 Universidad Carlos III de Madrid
%   Proyecto final de carrera de Rubén Darío Crespo Sánchez.</p>
%   Tutor: Ramón Barber Castaño.</p>
%   Octubre 2011.
%

% Get unique identifier (to form file name)
mlid = getfield(instruct, 'mlid');

% Set directory path for storage of graphic files.
cd(instruct.mldir);

% Cleanup jpegs over an hour old.
wscleanup('websimml*.jpeg', 1);
```

```
% Initialize the return string.
retstr = char('');

% Get the tiempo (string) variable. Convert to a number.
if(~length(instruct.tiempo))
  tiempo = 10; % Default empty field.
else
  tiempo = str2double(instruct.tiempo);% Extracción del parámetro tiempo
end

% Save size as a char string in structure OUTSTRUCT.  The
% function HTMLREP requires the first argument to be a structure
% containing the variables referenced in the output template
% file, websim2b.html.

load_system('modelo_simulink');
sim('modelo_simulink', tiempo); % Ejecución del modelo de simulink

t=ScopeData(:,1);
y=ScopeData(:,2);

%-------------------------------------------------------------
% Create the plot
%-------------------------------------------------------------

% setup to create a jpeg file
Fig = figure('visible','off');

plot(t,y);  % Gráfica con los datos
ylabel('Salida');
xlabel('Tiempo');

% Adjust size
```

```
pos = get(gcf, 'position');
pos(3) = 380;
pos(4) = 310;
set(gcf, 'Position', pos, 'PaperPosition', [.25 .25 12 9]);

%------------------------------------------------------------
% Write the figure file
%------------------------------------------------------------
%Render jpeg and write to file.
drawnow;
s.GraphFileName = sprintf('websim%s.jpeg', mlid);
wsprintjpeg(Fig, s.GraphFileName);
s.GraphFileName = sprintf('/icons/websim%s.jpeg', mlid);
close all;

% Put name of graphic file into HTML template file.
templatefile = which('websim2b.html');
retstr = htmlrep(s, templatefile);
```

Código de index.html

En este apartado se incluye el código del archivo index.html con la opción de elegir entre las 5 opciones desarrollados en este proyecto.

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<html>

<head>
<meta http-equiv="Content-Type"
content="text/html; charset=iso-8859-1">
<meta name="GENERATOR" content="Microsoft FrontPage 2.0">
```

```
<title>Laboratorios remotos mediante MATLAB Web Server y Easy Java Simulations</title>
</head>

<body bgcolor="#FFFFFF">

<p><font color="#000000" size="4" face="Arial"><i><img
src="/icons/logo_uc3m.gif" width="100" height="100"></i></font> <font
size="5"><strong><b>Laboratorios remotos mediante MATLAB Web Server y Easy Java Simulations</b></strong></font> </p>

<ul>
  <li><a href="websim1a.html">Ejecución de modelo_simulink.mdl (websim1a)</a>
    Ejecuta un fichero de simulink y devuelve los datos de
      la simulación.<font face="Arial Narrow"></font> Los resultados se muestran
      automáticamente en una tabla mediante MATLAB Web Server.</li>
</ul>

<ul>
  <li><a href="websim2a.html">Ejecución de modelo_simulink.mdl (websim2a)</a>
    Ejecuta un fichero de simulink y devuelve los datos de
      la simulación.<font face="Arial Narrow"></font> Los resultados se muestran
      automáticamente en una gráfica mediante MATLAB Web Server.</li>
</ul>

<ul>
  <li><a href="websim3a.html">Ejecución de modelo_simulink_2.mdl (websim3a)</a>
    Ejecuta un fichero de simulink y devuelve los datos de
      la simulación. Se pueden establecer los valores del numerador y denominador
      de la función de transferencia.<font face="Arial Narrow"></font> Los resultados se muestran
      automáticamente en una gráfica mediante MATLAB Web Server.</li>
</ul>
```

```
<ul>
  <li><a href="websim4a.html">Ejecución del motor (websim4a)</a>
    Ejecuta un fichero de MATLAB que se conecta con el motor y devuelve los datos del
      ensayo.<font
    face="Arial Narrow"></font> Los resultados se muestran
      automáticamente en una gráfica mediante MATLAB Web Server.</li>
</ul>

<ul>
  <li><a href="ejs_group.html">Abre una página con los ejemplos de Easy Java Simulations</a>
    Existen dos opciones: Ejecutar el modelo sencillo de simulink_interno.mdl o ejecutar el
      modelo simulink_interno_daq.mdl que se conecta con el motor.<font face="Arial Narrow">
      </font> Los resultados se muestran automáticamente en una gráfica mediante Easy Java
      Simulations.</li>
</ul>

<p><font color="#000000" size="4" face="Arial"><i><center> <img
src="/icons/maqueta.jpg" width="324" height="216"></center> </i></font>

<p><font color="#0000FF" size="2" face="Arial">©2011 Universidad
Carlos III de Madrid, Proyecto final de carrera de Ingeniería Industrial de Rubén Darío Crespo Sánchez.</p>
Tutor: Ramón Barber Castaño.</p>
Octubre, 2011.</font></p>
</body>
</html>
```

Código de websim2a.html

En este apartado se incluye el código del archivo websim2a.html como ejemplo de código de los demás ficheros. La estructura es muy similar en los demás archivos de websim#a.html.

```
<html>
<head>
<title>Ejecuta un fichero de simulink al pulsar Ejecutar</title>
</head>

<body bgcolor="#FFFFFF">
<p><font color="#000000" size="4" face="Arial"><i><img
src="/icons/logo_uc3m.gif" width="100" height="100">
</i></font></p>

<p><font color="#000000" size="4" face="Arial">
<i>Ejecuta el siguiente modelo de simulink al pulsar Ejecutar</i></font> </p>

<p><font color="#000000" size="4" face="Arial"><i><img
src="/icons/modelo_simulink.gif" width="512" height="158">
</i></font></p>

<!--
MATLAB Web Server requires only two items in all input HTML
documents:

    1. The form action must refer to the matweb program
    (matweb.exe on NT).

    2. A hidden field containing the name "mlmfile" and
    the value of the name of your application entry point
    function.  For example, "webmagic" in the hidden field
```

```
        below invokes the M-file webmagic.m.  (Note that there
        must be an entry in matweb.conf with the same name.
        See the file matweb.conf in the wsdemos directory
        and the documentation.)
-->
<form action="/cgi-bin/matweb.exe" method="POST">
   <input type="hidden" name="mlmfile" value="websim2">

   <p>Tiempo:
   <input type="text" size="5" maxlength="10" name="tiempo"></p>



   <p><input type="submit" name="Ejecutar" value="Ejecutar"></p>
</form>

<p> </p>

<p><font color="#0000FF" size="2" face="Arial">©2011 Universidad
Carlos III de Madrid, Proyecto final de carrera de Rubén Darío Crespo Sánchez.</p>
Tutor: Ramón Barber Castaño.</p>
Octubre, 2011.</font></p>
</body>
</html>
```

Código de websim2b.html

En este apartado se incluye el código del archivo websim2b.html como ejemplo de código de los demás ficheros. La estructura es muy similar en los demás archivos de websim#b.html.

```
<!--
websim2.html
```

```
-->

<html>
<head>
<title>Valores de la simulación obtenida de simulink</title>
</head>

<body bgcolor="#FFFFFF">
<p><font color="#000000" size="4" face="Arial"><i><img
src="/icons/logo_uc3m.gif" width="100" height="100"></i></font></p>

<div align="center">
<strong>Gráfica de la simulación obtenida de simulink</strong></p>

<p align="center">
<img border=0 src="$GraphFileName$">
</p>

<p>
<p>
<font color="#0000FF" size="2" face="Arial">©2011 Universidad
Carlos III de Madrid, Proyecto final de carrera de Rubén Darío Crespo Sánchez.</p>
Tutor: Ramón Barber Castaño.</p>
Octubre, 2011.</font></p>
</body>
</html>
```